Fig. 5.2.28 Effect of transmission errors in DPCM coders (a) Previous element prediction. (b) Error patterns for previous element prediction. (c) Spatial average prediction. (d) Error patterns for spatial average prediction.

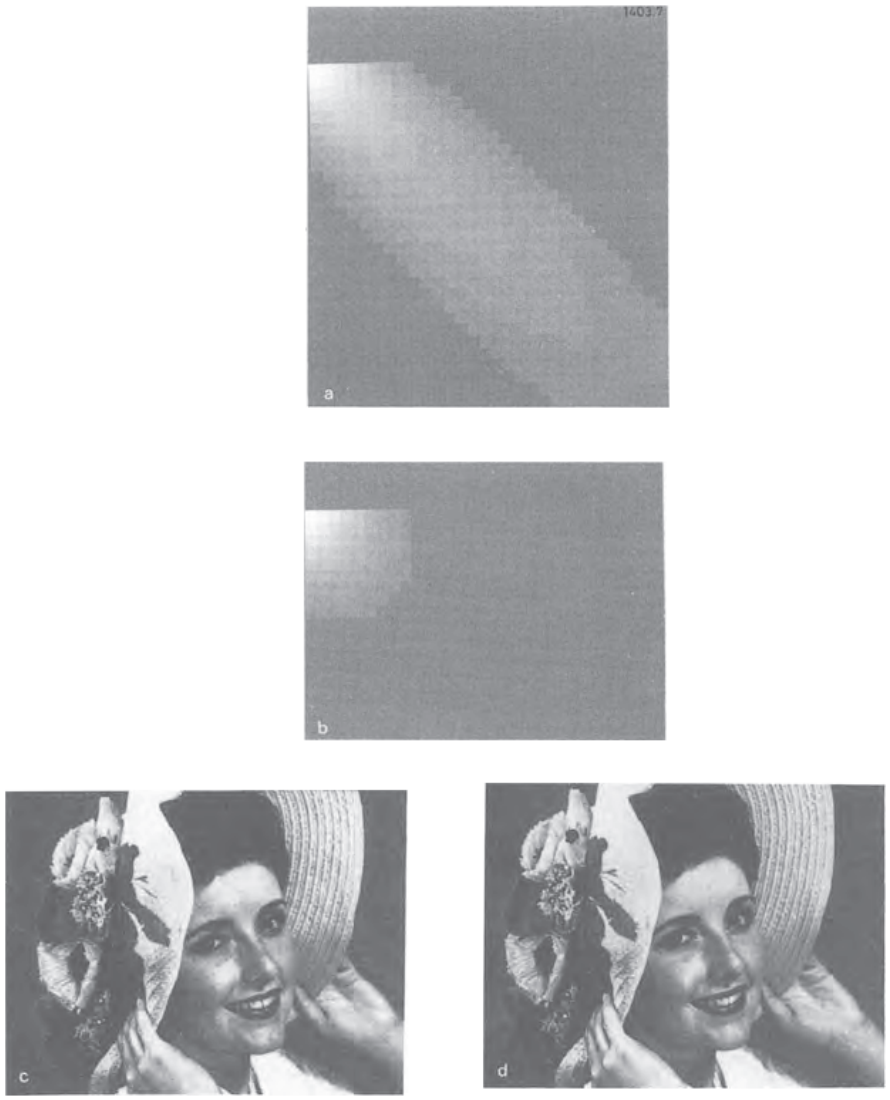


Fig. 5.2.29 (a),(b) Error patterns for the two-dimensional predictions of Eq. (5.2.55) and (5.2.55) in areas of uniform luminance; (c) and (d) are reconstructed pictures with channel error rate of 5×10^{-4}. (From Jung and Lippman [5.2.17])

5.2.3k Effects of Transmission Errors

One important aspect in the design of predictors is the effect of transmission bit errors in the reconstructed picture. In a DPCM coder, the prediction of each pel is made using certain previously transmitted pels. Thus, a transmission bit error in any pel could affect one or more future pels, which in turn could affect all of the subsequent pels. The amount and spread of this error propagation depends both upon the type and the functional definition of the predictor used. In the case of the optimum linear one-dimensional predictor for zero mean pels, it is known that for most types of correlations generally found in pictures, the predictor coefficients are such that the effect of transmission error decays as desired.

The previous element predictor with unity weighting coefficient does not limit the effects of transmission errors. This is clearly seen by an example. Let us assume that the quantizer used in the previous element DPCM coder maps input into output exactly. Then as shown in Fig. 5.2.27, a transmission error of 10 at pel number 5 gets propagated for all the subsequent pels along the scan line. In real coders, this appears as a horizontal streak in the reconstructed picture. However, the propagation errors can be limited by providing a *leak*, i.e., using a weighting coefficient less than one. In this case a fixed fraction of the previous element value is used for prediction. Fig. 5.2.27 also shows that with the leak, the transmission error has been reduced to zero after pel number 12. The rate of decay is determined by the predictor coefficients.

For two dimensional predictors the situation is more complex. The optimum* linear two dimensional predictors for most commonly used correlation functions may be unstable, i.e., transmission errors may propagate throughout the picture. Good criteria for stability of predictors are not yet known. However, there are some stable two-dimensional predictors for which the pattern of distortion produced by transmission errors is much less annoying than with one dimensional predictors. Fig. 5.2.28 shows error patterns produced by a previous element and a two dimensional predictor. Fig. 5.2.29 shows the propagation of transmission error for two different predictors. The predictors are: (using pel positions in Fig. 5.2.8)

$$\text{predictor I} \quad \triangleq 3/4\,A - 1/2\,B + 3/4\,C \tag{5.2.55}$$

$$\text{predictor II} \quad \triangleq 3/4\,A - \frac{17}{32}\,B + 3/4\,C$$

* Optimum in the sense of minimum mean square prediction error.

Fig. 5.2.29(a) and (b) show response of the DPCM receiver to an impulse of transmission error in a pel at the upper left hand corner of the picture in an area of uniform luminance, for the two predictors. It is clear that rate of decay in the first predictor is slow since the sum of predictor coefficients is equal to one, whereas for the second predictor rate of decay is much faster, since the sum of predictor coefficients is less than one. Fig. 5.2.29(c) and (d) show the corresponding reconstructed pictures. It is not easy to perceive the differences between these two pictures due to loss of quality in the printing process, but on a real television monitor the differences are quite obvious.

In the case of adaptive predictors, a transmission error can have two deleterious effects - one due to incorrect values for the prediction pels, and the other due to selecting the wrong predictor at the receiver. Some improvement in adaptive predictor performance may be obtained by providing a leak in the rule of adaptation, i.e., by using a fraction of the adaptive prediction and a fixed predictor. However, in general, the criteria for stability of such predictors and the effects of transmission bit errors are much less understood. The only recourse may be the use of error detecting and correcting codes.

In interframe coding, the effect of transmission errors may last for several frames, and in most cases it is necessary to employ techniques of error correction and/or concealment. Several techniques have been investigated. These are in addition to the use of error detecting/correcting codes that may be used for very noisy channels. Two commonly used techniques are: (a) hybrid PCM/DPCM and (b) use of Remaining Redundancy. In hybrid PCM/DPCM, predictor coefficients are reset to zero periodically and during the resetting a PCM value is transmitted. A commonly used scheme is to transmit a PCM value for the first pel of every M-th line. This limits every transmission error impulse to M lines. The second technique utilizes the redundancy present in the picture signal to restore the error-corrupted DPCM-decoder output. Such techniques first attempt to detect transmission errors either by error detecting codes or by observing anomalies in the reconstructed picture at the receiver. If an error is detected, then some previously transmitted part of the picture (which is assumed to be error free) is repeated or averaged. One commonly employed scheme is to either repeat a previous scan line or a scan line from the previous frame at the same location. Better performance is obtained by replacing the error corrupted line by an average of some previous and future lines. Reliable detection of errors is important in these techniques, otherwise due to repeating or averaging the scan lines, there can be substantial loss of resolution.

5.2.4 Quantization

DPCM schemes achieve compression, to a large extent, by not quantizing the prediction error as finely as the original signal itself. Several methods of optimizing quantizers have been studied, but quantizer design still remains an art and somewhat ad hoc. Most of the work on systematic procedures for quantizer

optimization has been for the previous-element DPCM coder, in which the approximate horizontal slope of the input signal is quantized. Although obvious extensions can be made to the case of two-dimensional and interframe predictors, they have not yet been thoroughly studied. For this reason, we shall discuss in detail quantizers for the previous-element DPCM and point out how extensions to the other cases can be made.

Three types of degradations may be seen as a result of DPCM quantization. These are referred to as *granular noise*, *slope overload* and *edge busyness* and are shown in Fig. 5.2.30. If the inner levels (corresponding to the small magnitudes of the differential signal) of the quantizer are too coarse, then the flat or low detail areas of the picture are coarsely quantized. If one of the output levels is zero (i.e., a mid-step quantizer), then such coarse quantization may show false contours in the reconstructed picture. Otherwise, it has the appearance of random or granular noise added to the picture. On the other hand, if the dynamic range (i.e., largest representative level) of the quantizer is too small, then for every high contrast edge, the decoder takes several samples for the output to catch up to the input, resulting in slope overload, which appears similar to low-pass filtering of the image. Finally, for edges whose contrast changes somewhat gradually, the quantizer output oscillates around the signal value and may change from line to line, or frame to frame, giving the appearance of a discontinuous jittery or "busy" edge.

Quantizers can be designed purely on a statistical basis or by using certain psychovisual measures. They may be adaptive or fixed. We shall discuss each of these cases in the following sections.

5.2.4a Nonadaptive Quantization

Optimum quantizers that are based on statistics have been derived using the minimum mean square error criterion.[5.2.18] Considering Fig. 5.2.31, if x with probability density $p(x)$, is the input to the DPCM quantizer, then quantizer parameters can be obtained to minimize the following measure of quantization error:

$$\mathbf{D} = \sum_{k=1}^{L} \int_{t_{k-1}}^{t_k} f(x - \ell_k) \cdot p(x)\, dx \tag{5.2.56}$$

where $t_0 < t_1 < \ldots < t_L$ and $\ell_1 < \ell_2 < \ldots < \ell_L$ are decision and representative levels, respectively, and $f(\cdot)$ is a nonnegative error function. We assume, as shown in Fig. 5.2.31 that all inputs $t_{k-1} < x \leq t_k$ to the quantizer are represented as ℓ_k. If one of the representative levels is zero then the quantizer is termed *mid-tread,* and if one of the decision levels is zero then it is called *mid-riser*. Necessary conditions for the optimality with respect of t_k and ℓ_k for a fixed number of levels L are given by:

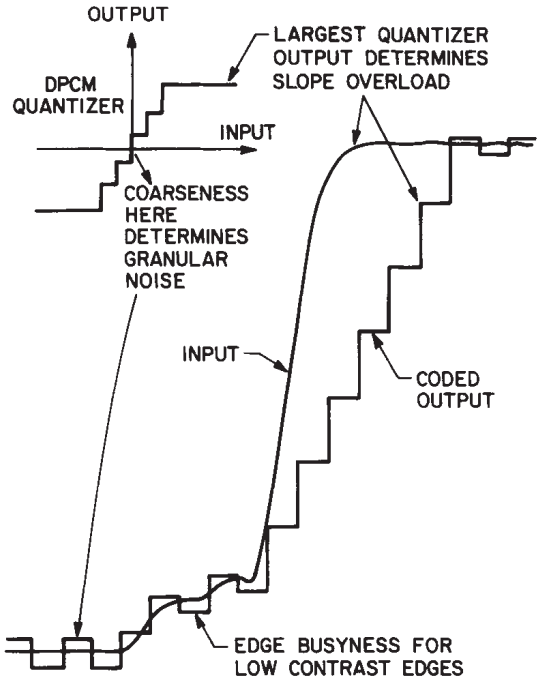


Fig. 5.2.30 An intuitive classification of quantizing distortion due to DPCM coding. Three classes of quantization noise are identified: granular noise, edge busyness and slope overload.

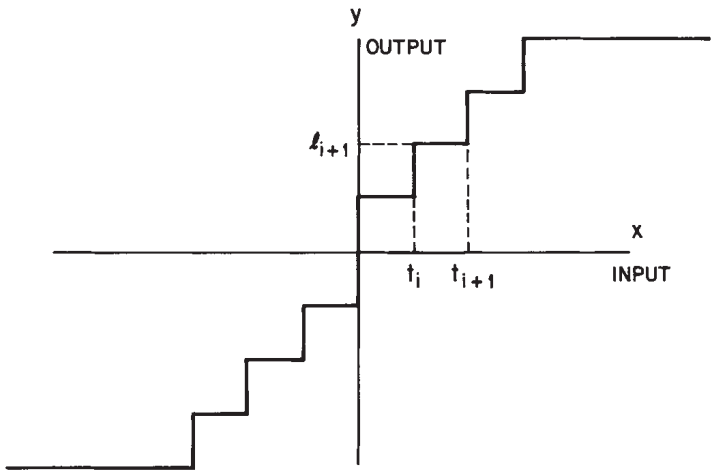


Fig. 5.2.31 Characteristics of a quantizer. x is the input and y is the output. $\{t_i\}$ and $\{\ell_i\}$ are decision and representative levels, respectively. Inputs between t_i and t_{i+1} are represented by ℓ_{i+1}.

$$f(t_{k-1} - \ell_{k-1}) = f(t_{k-1} - \ell_k), \quad k = 2 \ldots L \tag{5.2.57}$$

and

$$\int_{t_{k-1}}^{t_k} \frac{df(x - \ell_k)}{dx} \cdot p(x)\,dx = 0, \quad k = 1 \ldots L \tag{5.2.58}$$

assuming that $f(\cdot)$ is differentiable. In the case of the mean-square error criterion, $f(z) = z^2$, the above equations reduce to the Lloyd-Max quantizer[5.2.18] (see also Eq. (3.2.30))

$$t_k = 1/2\ (\ell_k + \ell_{k+1}) \tag{5.2.59}$$

and

$$\ell_k = \int_{t_{k-1}}^{t_k} xp(x)\ dx \ / \int_{t_{k-1}}^{t_k} p(x)\,dx. \tag{5.2.60}$$

Several algorithms and approximations exist for the solution of these equations. The probability density $p(x)$ for the case of previous pel and other differential coders can be approximated by a Laplacian density (see Chapter 3), and the optimum quantizer in this case is *companded*, i.e., the step size $(t_k - t_{k-1})$ increases with k. Since much of the time the picture signal has slope x close to zero, this results in a fine spacing of levels near $x = 0$ and a rather coarse spacing of levels as $|x|$ increases. For images, this generally results in overspecification of the low-detailed areas of the picture, and consequently only a small amount of granular noise, but relatively poor reproduction of detailed areas and edges. Due to the small dynamic range of such a quantizer, this is often visible as slope overload and edge busyness for high contrast, vertical edges. Thus for subjective reasons, the quantizers should not be designed based on the *MSE* criterion.

Minimization of a quantization error metric, i.e., distortion, for a fixed number of levels is relevant for DPCM systems that code and transmit the quantizer outputs using fixed-length binary words, since in that case the output bit-rate depends only on the logarithm of the number of levels of the quantizer. However, since the probability of occurrence of different quantizer levels is highly variable, there is a considerable advantage in using variable length words to represent quantizer outputs. In such a case, since the average bit-rate (in bits/pel) is lower bounded by the entropy of the quantizer output, it is more relevant to minimize the quantization distortion subject to a fixed entropy. It was stated in Chapter 3 that for Laplacian random variables and a mean square error distortion criterion, the optimum quantizers are uniform. Although such

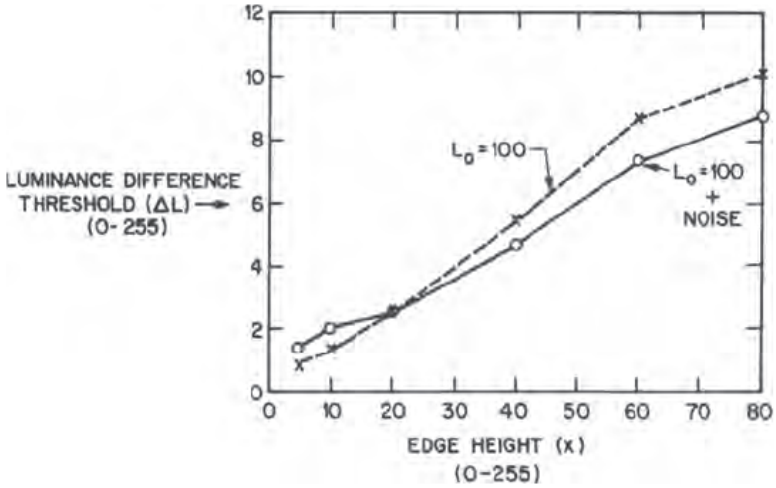


Fig. 5.2.32 Visibility threshold versus slope for a single luminance edge that rises to its peak value within one pel. Edge height and luminance difference threshold are plotted in units 1 to 255 (8-bit number) for two cases: 1) without any additive noise, and 2) with additive noise. (from Sharma and Netravali [5.2.19])

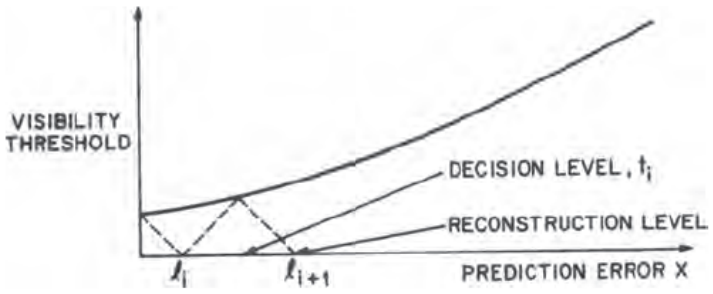


Fig. 5.2.33 Illustration of the quantizer design procedure for an even number L of reconstruction levels.

quantizers may provide adequate picture quality for sufficiently small step size, they may often be too complex to implement because of the large number of levels and the very long word-lengths required for the outer levels. Simplifications are necessary to keep the complexity small.

It has been realized for some time that for better picture quality, quantizers should be designed on the basis of psychovisual criteria. However, a debate continues on what criterion should be used, and expectedly so, considering the complexities of the human visual system. We will describe two approaches and point the reader to other works.

In the first method, a quantizer is designed such that the quantization error is at or below the threshold of visibility (using a given model of the error visibility) while minimizing either the number of quantizer levels or the entropy of the quantizer output. In Chapter 4 it was pointed out that the visibility of perturbations in image intensity is masked by spatial variations in intensity. The visibility threshold at an edge is the value by which the magnitude of the edge (i.e., the edge contrast) can be perturbed such that the perturbation is just visible. A typical relationship[5.2.19] is given in Fig. 5.2.32 for an edge width of one pel and a background luminance L_o = 100 (on a scale of 0-255) and variable height. Knowing this relationship between the edge visibility threshold and the slope of a single edge, the quantization error can be constrained to be below this threshold for all slopes of the signal. The algorithm for defining the characteristics of the quantizer satisfying this constraint is illustrated in Fig. 5.2.33, for an even L (number of quantizer output levels). In this case, $x = 0$ is a decision level, (i.e., mid-riser) and the geometric procedure starts at the corresponding point on the visibility threshold function with a dashed line of inclination -45°. The intersection of this line with the abscissa gives the first reconstruction level. From this point, a dashed line of inclination +45° is drawn. Its intersection with the visibility threshold function gives the next decision level, and so on. This procedure is continued until the last +45° line exceeds the amplitude range of x. The number of points at which this path meets the abscissa is the minimum number of levels of the desired quantizer. The resulting dashed zigzag line represents the magnitude of the quantization error as a function of the quantizer input signal x. If L is odd, (i.e., mid-tread) then the procedure starts with the reconstruction level at $x = 0$.

Experimental evidence indicates that quantizers satisfying the above threshold constraint turn out to be the same whether the number of levels is minimized or the entropy is minimized. For videotelephone type (i.e., 256×256 pels with 1 MHz bandwidth) monochrome signals with head and shoulders types of pictures, 27 levels are required for no visible error at normal (six times the picture height) viewing distance if previous element prediction is used. A larger number of levels are required for pictures containing a lot of sharp detail, e.g. the resolution charts. Using two dimensional prediction, this number may be reduced to 13 levels for natural (e.g. head and shoulders) pictures and 21 levels

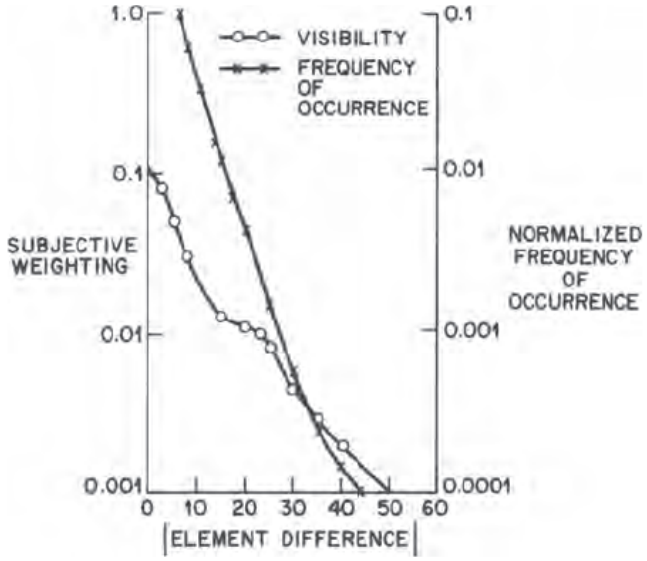


Fig. 5.2.34 Subjective weighting function for quantization noise visibility as related to the magnitude of the element difference for a typical head and shoulders view. Also shown is a histogram of the magnitude of the element differences. (from Netravali [5.2.20])

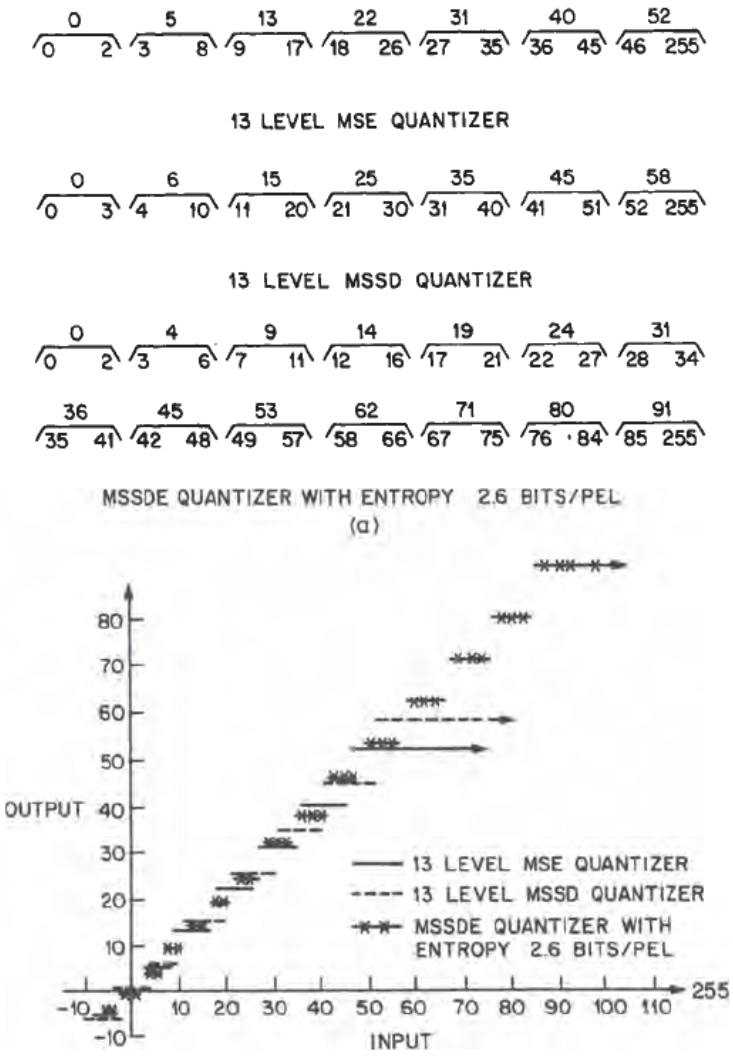


Fig. 5.2.35 Input and output characteristics of quantizers optimized under three criteria are shown: (1) 13 level minimum mean square error (MSE); (2) 13 level minimum subjective weighted mean square error (MSSD); (3) MSSD with a constraint on the entropy (MSSDE) of the quantizer output. For (2) and (3), weights are derived from subjective weighting functions such as in Fig. 5.2.34. (from Netravali [5.2.20])

for resolution charts. Chrominance components usually require fewer levels, e.g., 9 levels for $(R-Y)$ and 5 for $(B-Y)$.

A second method of designing psychovisual quantizers is to minimize a weighted square quantization error, where the weighting is derived from subjective tests. Such optimization would be similar to the MMSE quantization, where the probability density is replaced by a subjective weighting function obtained either from the threshold data (e.g., reciprocal of the threshold data, properly scaled) or from direct subjective tests using complex pictures as stimuli. A typical subjective weighting function for a head and shoulders type of picture is shown in Fig. 5.2.34 when a previous pel predictor is used.[5.2.20] Such weighting functions are picture dependent, but the variation for a class of pictures (e.g., head and shoulders views) is not significant. There has been and perhaps will continue to be considerable debate on the choice of the subjective weighting functions. Although the optimum choice of the weighting function is unknown, optimum quantizers designed on the basis of the above weighting functions have been quite successful. In general, quantizers that minimize the subjective weighted square error for a fixed number of levels (MSSD) or fixed entropy (MSSDE) are less companded than the minimum mean-square error (MMSE) quantizers, and reproduce edges more faithfully. Fig. 5.2.35 and Fig. 5.2.36 give characteristics of quantizers optimized under different criteria and the performance of these quantizers in terms of entropy of the quantizer output and the MSSD measure of picture quality.

Quantizer design for intraframe coders that use more sophisticated predictors has been either on the basis of MMSE or by trial and error. However, the methods of psychovisual quantizers discussed above can be extended easily for this case. For interframe coders, the situation is somewhat more complicated since the quantization error occurs predominantly in the moving areas of the picture, and its visibility depends upon the spatial as well as temporal variations in the scene. Because of the lack of suitable models for both the distribution of the prediction error and the visibility of quantizing error, the problems of designing both MMSE and psychovisual quantizers have not been studied to a great extent. Quantizers used in practice are based on trial and error.

In most applications, the picture signal is positive and has a limited range (say from 0 to 255). If in a DPCM system, the prediction of a pel is $\hat{b}$, then the range of possible inputs to the quantizer is $(-\hat{b}, 255 - \hat{b})$. Thus, unless $\hat{b} = 128$, several levels of a symmetric quantizer may not get used. For example, if $\hat{b} = 253$, the only positive inputs to the quantizer will be 1 and 2, and therefore the quantizer levels corresponding to the prediction errors above 2 will be wasted. To overcome this, a "prediction-value-dependent" mapping of the quantizer characteristic can be designed. If the quantizer is implemented as a look-up-table (using random access memories), then several look-up-tables can be designed and the appropriate one can be selected depending upon the value of the prediction.

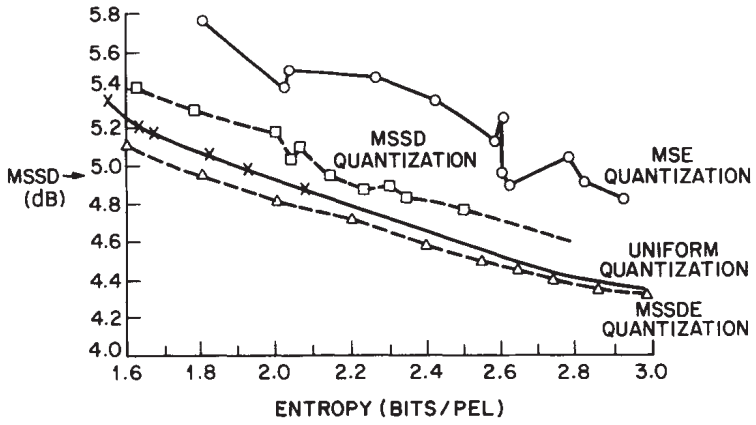


Fig. 5.2.36 Performance of quantizers optimized under different criteria for a typical head and shoulders picture. Mean-square subjective weighted quantization noise (MSSD) is plotted with respect to entropy. It is assumed that MSSD is a reasonable measure of picture quality. (from Netravali [5.2.20])

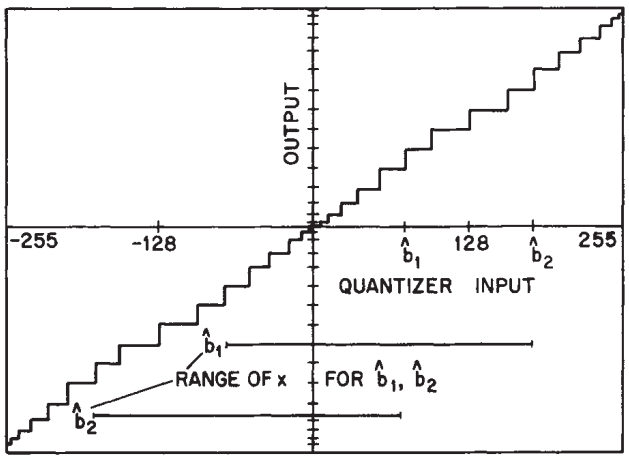


Fig. 5.2.37 Quantizing characteristics of a reflected quantizer.

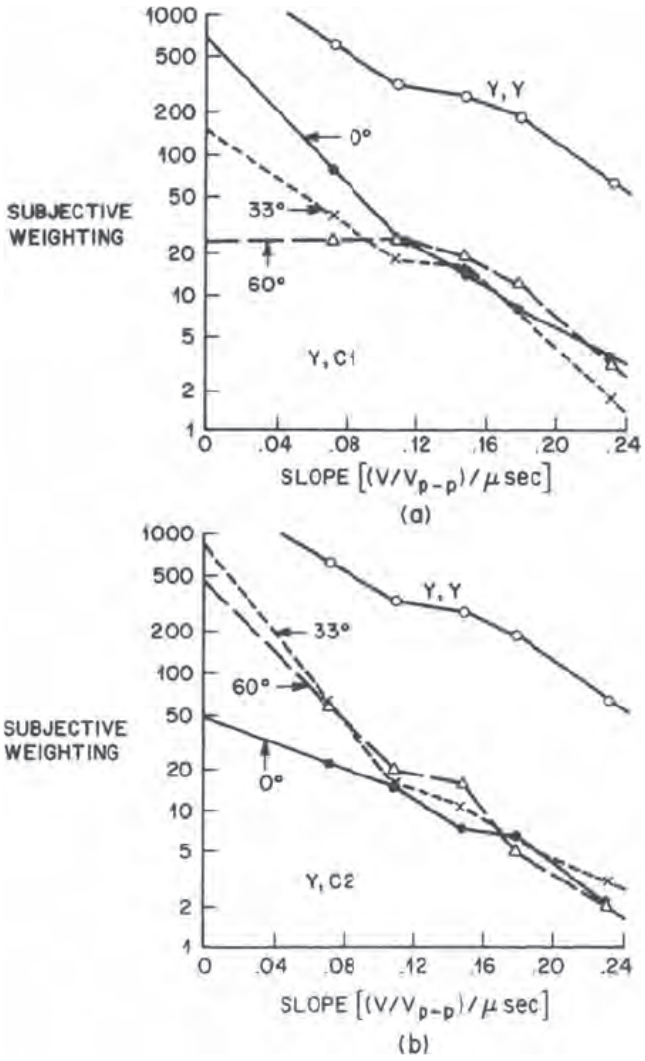


Fig. 5.2.38 Subjective weighting functions with thresholding of luminance slope. This corresponds to using previous element DPCM. Noise resembling quantization noise is added to Y (Y,Y curve), and to (a) $C1$ and (b) to $C2$ for different values of θ as shown. Abscissa is luminance slope measured as a ratio of voltage difference to peak voltage per unit time. Typical videotelephone standard was used, i.e., bandwidth of luminance and two chrominance components was 1 MHz and 250 kHz, respectively. (from Netravali and Rubinstein [5.2.21])

A *Reflected* quantizer[5.2.6] accomplishes this to some extent, first, by representing the prediction error in two's complement arithmetic so that same binary codeword represents values x and $256-x$. Next, a special quantizer characteristic shown in Fig. 5.2.37 is used. The range of possible prediction errors then covers a fixed number of levels with a unique set of code words for any given value of the prediction. Fig. 5.2.37 shows ranges of prediction error x for two values $\hat{b}_1$ and $\hat{b}_2$ of the prediction. In this example, 64 quantizing levels can be represented by only 5 bits, saving 1 bit per pel. Although there are other prediction-value-dependent mappings of the quantizer characteristics, the reflected quantizer is particularly simple to implement. Unfortunately, it has found only limited use since it forces the same step size for both small and large prediction errors, which is not consistent with desired companding characteristics based on psychophysics.

5.2.4b Nonadaptive Quantization for Color Signals

When the color signal is represented as luminance and chrominance components, each of them can be coded using a DPCM quantizer. In this case, procedures for either statistical optimization (MSE) or subjective optimization (MSSD) of the quantizer characteristics are the same as those outlined in the previous section for the luminance signal. However, the subjective data required for quantizer optimization is not yet available in many cases. Figs. 5.2.38 and 5.2.39 show subjective noise weighting functions for the color components when the previous pel predictor is used.[5.2.6] As in the case of monochrome pictures, such data are picture dependent, but they can be used for a class of pictures by minimizing the mean square subjective distortion (MSSD) to obtain the DPCM quantizers.

Weighting functions in these figures are given for three values of θ, where θ represents the rotation of the chrominance axes in the $(R-Y)/1.14$, $(B-Y)/2.03$ plane. $\theta = 33°$ corresponds to the NTSC I and Q signals. Fig. 5.2.38 gives the subjective weighting functions of noise added to the chrominance signal at those pels where the normalized horizontal slope of the luminance equals the abscissa. The top curve in each case shows the visibility of noise in the luminance as a function of the luminance slope. The weighting function for the luminance signal is much greater than that for the chrominance signal. This implies, as expected, that fewer levels are required for the chrominance quantization. The luminance weighting curve is used the same way as in the previous sections. Chrominance weighting plotted as a function of luminance slope can be used for adaptive quantization of chrominance and is discussed in Section 5.2.4c. Fig. 5.2.39 gives similar results with thresholding of the chrominance slope. Here chrominance noise weighting is plotted as a function of chrominance slope, and therefore these weighting functions can be used to derive MSSD quantizers as discussed in the previous section.

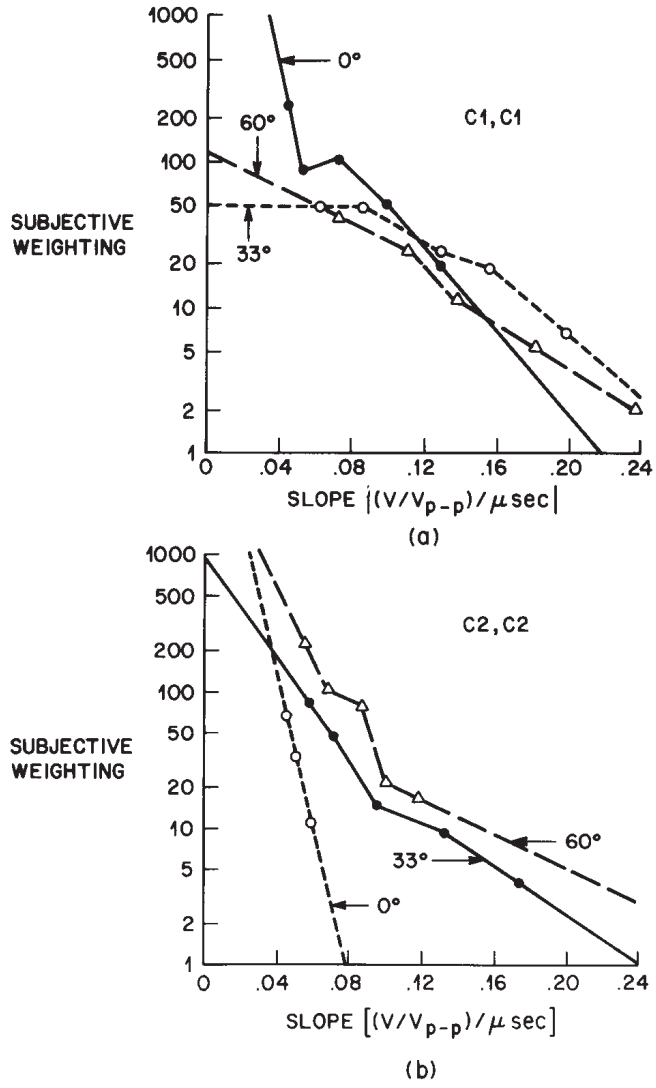


Fig. 5.2.39 Subjective weighting functions with chrominance slope thresholding. (a) Noise resembling quantization noise is added to $C1$ as a function of $C1$ slope and (b) to $C2$ as a function of $C2$ slope for different values of θ. (from Netravali and Rubinstein [5.2.21])

A comparison of Fig. 5.2.38 and Fig. 5.2.39 shows that in general the subjective weighting as a function of the luminance slope is lower than as a function of chrominance slope. This implies that a luminance slope provides a better masking of the noise in the chrominance signal than does the chrominance slope. This added masking due to the luminance is a result of limitations of visual perception, as well as the unequal filtering of the luminance and chrominance signals. It is used in adaptive quantization of the chrominance signal (see next section). It can also be observed from Fig. 5.2.39 that for most of the slope values, noise in $C1$ has higher visibility than noise in $C2$ for $\theta = 0°$ and $\theta = 33°$, with $\theta = 60°$ exhibiting somewhat equal visibility between the two chrominance components. The greater visibility for $C1$ noise implies that finer quantization of the $C1$ signal is necessary compared to the $C2$ signal.

These subjective weighting functions can be used to derive the MSSD quantizers with the help of Eqs. (5.2.57) and (5.2.58). Table 5.2.2 shows results of this type of quantizer design as well as the quantizers designed on the basis of minimum mean square error (MSE) for a fixed number of levels using the statistics of a head and shoulders type of image. As in the case of luminance signals (previous section) such subjective quantizers are much less companded than the MSE quantizers. MSE quantizers show over-design in the flat areas of the picture where the signal slope is small. They also have low dynamic range, which causes annoying slope overload and a consequent decrease in the resolution of the coded pictures. Also, because the inner levels of the *MSE* quantizers are more closely spaced than those of the subjective quantizer, the probability distribution of the quantized output of the *MSE* quantizer is more uniform. This results in the *MSE* quantizer output having higher entropy than the output of the subjective quantizer, for the same number of levels.

5.2.4c Adaptive Quantization

Due to the variation of the image statistics and the required fidelity of reproduction in different regions of the picture, adapting the DPCM quantizer is advantageous in terms of reducing the bit rate. In general, one would like to segment a picture into several subpictures such that within each subpicture both the quantization noise visibility and the statistical properties of the quantized differential signal are relatively stationary. However, this is an extremely difficult task since the perception of noise and the statistics may not be sufficiently related to each other. Several approximations of this idealized situation have been made, some purely statistical and some based on certain psychovisual criteria. For example, one can work in the spatial frequency domain and split the signal into two frequency bands in order to exploit the sensitivity of the eye to variations in image detail (see Chapter 4). In this case, the signal in the low-frequency band is spatially sampled at a low rate, but is quantized finely because of the high sensitivity of the eye to the noise in the low frequency region. The high frequency component of the video signal is spatially

Table 5.2.2 Quantizer Characteristics for DPCM Coding of Color Components. Picture is segmented in 2 or 4 segments and uniform quantizer of different step size is used for each segment.

		NO. OF LEVELS	ONE SIDED QUANTIZER CHARACTERISTIC (ON A SCALE OF 0-255)						
LUMINANCE CODING	VISIBILITY	13	0	3	8	15	22	31	44
			0 1	2 5	6 11	12 18	19 26	27 36	37 255
	MMSE	13	0	2	5	10	15	20	25
			0 1	2 3	4 7	8 12	13 17	18 22	23 255
CHROMINANCE CODING C1	VISIBILITY	9	0	4	10	18	37		
			0 2	3 7	8 14	15 27	28 255		
	MMSE	9	0	2	8	18	29		
			0 1	2 5	6 13	14 23	24 255		
CHROMINANCE CODING C2	VISIBILITY	7	0	3	6	12			
			0 1	2 4	5 9	10 255			
	MMSE	7	0	2	6	10			
			0 1	2 3	4 7	8 255			

* NOTATION $\overbrace{}^{z}$ (x y) INDICATES THAT SIGNAL BETWEEN x AND y (INCLUDING x, y) IS QUANTIZED AS z.

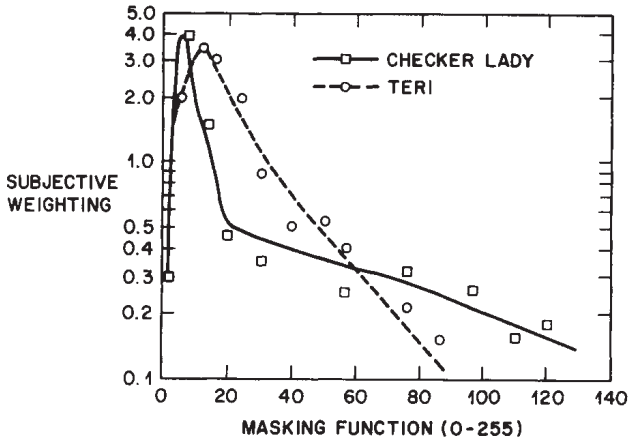


Fig. 5.2.40 Relationship between subjective visibility of noise and a measure of spatial detail (called the *Masking function*) that uses a combination of element and line differences in a 3×3 neighborhood. Dependence of the visibility on the picture content is shown for two head-and-shoulders views. (from Netravali and Prasada [5.2.22])

sampled at a higher rate, but is quantized coarsely due to reduced sensitivity to noise in highly detailed regions.

Pel domain approaches to adaptive quantization start out with a measure of spatial detail (called the *Masking Function**) and then obtain experimentally the relationship between the visibility of noise and the measure of spatial detail. In nonadaptive quantization of the previous section, the prediction error is used as the measure of spatial detail, but for adaptive quantization more complex measures are used. A typical relationship for head and shoulders pictures is shown in Fig. 5.2.40, where a weighted sum of slopes in a 3×3 neighborhood is used as a measure of local spatial detail.[5.2.22] If $\Delta^H_{i,j}$ and $\Delta^V_{i,j}$ are the horizontal and vertical slopes of the luminance at location (i, j) then the weighted sum of slopes is given by

$$M(i\ j) = \sum_{\ell,\, m=-1}^{+1} \alpha^{\sqrt{\ell^2+m^2}} \left[|\Delta^H_{i-\ell,\, j-m}| + |\Delta^V_{i-\ell,\, j-m}| \right] \quad (5.2.61)$$

where (i, j) are the coordinates relative to the pel being coded and $\alpha < 1$ is a factor based on psychovisual threshold data. In the experimental investigation of [5.2.22], α is taken to be 0.35, and therefore, exponentially decreasing weights are assigned to pels of greater distance from the current (to be coded) pel. Using the noise weighting function of Fig. 5.2.40, which relates the visibility of additive noise to the masking function, a picture is divided into (not necessarily contiguous) segments. Within each segment noise visibility is approximately constant, and an individual quantizer is designed for each one. The segmentation is controlled by the value of the masking function M, as indicated in Fig. 5.2.41. Fig. 5.2.42 shows a head and shoulders picture divided into four segments. Fig. 5.2.42a shows an original picture, and Figs. 5.2.42b–e show four subpictures (pels in these subpictures are shown white) which are in the order of increasing spatial detail, i.e., decreasing noise visibility. Thus, flat low detail areas of the picture constitute subpicture I (i.e., Fig. 5.2.42(b)) and sharp, high contrast edges are included in subpicture IV (i.e., Fig. 5.2.42(e)). Quantizers for each of the subpictures could be designed in a way similar to the earlier nonadaptive techniques.

One side benefit of such an adaptive quantization scheme is that the prediction error (i.e., the differential signal) statistics show a marked change from one segment to the other. In Fig. 5.2.43, prediction error histograms are shown for the four subpictures of Fig. 5.2.42. It is evident that since these histograms are quite different from one another, a different variable length code

---

* We will later refer to this as an *Activity Function*.

Table 5.2.3 Performance of Adaptive Chrominance Coding.

NO. OF SEGMENTS	ADAPTIVE QUANTIZER STEP SIZES CHROMINANCE 1	CHROMINANCE 2	EQUIVALENT PICTURE QUALITY	ADVANTAGE OF ADAPTION (BITS/CHROMINANCE PEL)
2	2,7	3,10	A	0.16
2	3,10	4,12	B	0.21
2	3,8	4,12	B	0.18
2	4,14	5,15	C	0.27
2	2,7	4,12	A	0.26
2	3,10	5,15	C	0.11
2	5,10	7,12	D	0.19
4	2,3,5,20	2,4,7,13	A	0.35
4	2,6,10,20	3,7,10,20	A	0.40
4	3,4,7,19	4,7,14,20	B	0.31
4	3,5,8,20	4,9,14,20	B	0.38
4	3,8,12,20	4,10,14,20	B	0.43
4	4,6,10,20	4,7,14,20	B	0.48
4	4,6,10,20	5,8,14,20	C	0.37

NOTES: EQUIVALENT PICTURE QUALITY JUDGED USING NONADAPTIVE QUANTIZER STEP SIZE OF 2,3(A); 3,4(B); 3,5(C); 4,6(D).

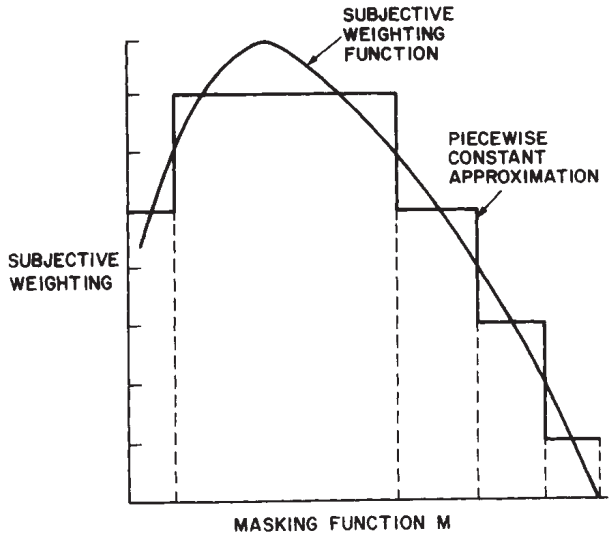


Fig. 5.2.41 Sketch of a typical subjective weighting function versus masking function M and its approximation by a four-level piecewise constant function. Using this approximation, images are segmented into four subpictures, I to IV, according to the value of the masking function M at each pel.

Fig. 5.2.42 Original picture (a) and its four segments (b), (c), (d), (e) designed on the basis of spatial detail (two-dimensional) and noise visibility. Segment (b) has the highest noise visibility, whereas segment (e) has the least noise visibility. (from Netravali and Prasada [5.2.22])

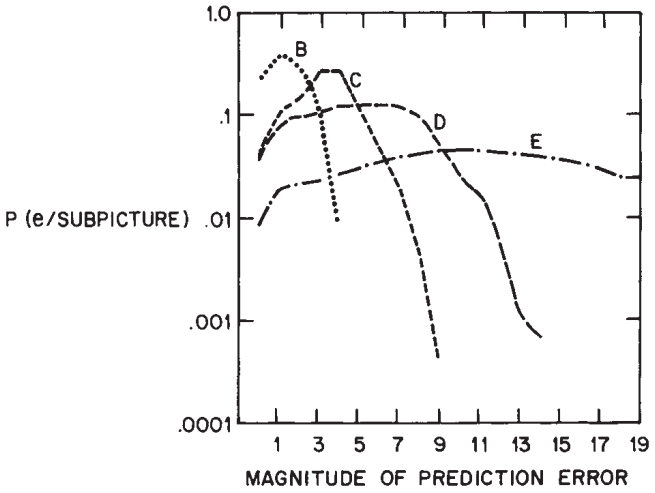


Fig. 5.2.43 Plot of conditional histograms of prediction errors conditioned on the four subpictures b, c, d, e as shown in Fig. 5.2.42. These subpictures are derived using a scheme similar to the one shown in Fig. 5.2.41. (from Netravali and Prasada [5.2.22])

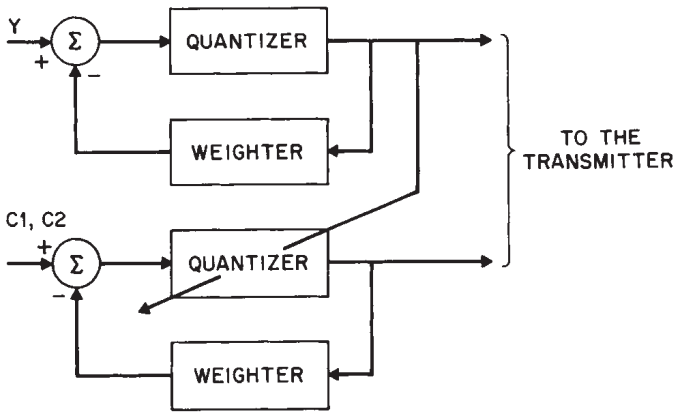


Fig. 5.2.44 Adaptive DPCM coding of the chrominance components. The chrominance quantizer is varied as a function of the luminance spatial activity as measured by the quantized luminance prediction error.

may be used to represent the quantizer outputs from each segment. This technique together with adaptive quantization is capable of reducing the bit-rate by about 30 percent compared with a nonadaptive previous pel DPCM coder. However, much of this reduction is a result of coarser quantization of different segments rather than the use of different variable word-length codes for different segments. It is also interesting to note that the advantage of adaptation increases with the number of segments used, up to about four, after which further improvement is small.

Another method of adapting the quantizer, called *delayed coding*, treats quantization as if following a path through an encoding tree determined by the quantizer steps. [See also Section 5.2.1 for a discussion on delayed coding in the context of delta modulation.] The path is chosen to minimize the weighted quantization error at several samples, including some subsequent samples that have yet to be transmitted. Since the representative levels are not varied from sample to sample, it is not necessary to send the adaptation information to the receiver. Thus the same input value to the quantizer may get mapped into a different representative level depending on which path on the tree will yield lower distortion. This is in contrast to the usual memoryless quantizer where every input value between the two decision levels is mapped into the representative level in between. As discussed in Section 5.2.1, such techniques have been used more successfully for delta modulation, since the growth of the tree is much smaller compared to DPCM with its many more quantizer levels.

In summary, for intraframe coding, although the optimum rules for adaptation are not known, rules based on intuition and trial-and-error have shown significant reduction in bit-rate compared with nonadaptive systems. In the case of interframe coders, systematic investigations of adaptive quantizers have yet to be made, primarily due to the lack of understanding of both statistics and the perceptual basis for adapting the quantizers. However, quantizers have been adapted in an ad hoc manner to facilitate matching of the variable bit-rate coder to the constant bit-rate channel. We shall discuss this in more detail in Section 6.6.

For coding of color components, since luminance edges provide better masking of the noise in the chrominance signal than do the chrominance edges (see Fig. 5.2.38), luminance edges should be used to compute the masking functions for adaptive coding of chrominance components. Fig. 5.2.44 shows a block diagram of a chrominance DPCM coder whose quantizer is varied as a function of the spatial activity of the luminance signal as measured by the quantized luminance prediction error. Experimental evidence indicates that by using the magnitude of the horizontal luminance slope as the measure of spatial activity, considerable advantage can be derived by adaptive coding. Table 5.2.3 summarizes the results. Here pictures coded with adaptive quantizers are subjectively compared to the nonadaptive quantizers. Differences in the entropies of the quantizer output for two (adaptive and nonadaptive) subjectively

LEVEL NO.	CODE WORD LENGTH	CODE
1	12	100101010101
2	10	1001010100
3	8	10010100
4	6	100100
5	4	1000
6	4	1111
7	3	110
8	2	01
9	2	00
10	3	101
11	4	1110
12	5	10011
13	7	1001011
14	9	100101011
15	11	10010101011
16	12	100101010100

Fig. 5.2.45 A typical variable-length code for a DPCM coded signal, with 16 quantizer levels. (from Pirsch [5.2.23])

equivalent pictures are tabulated. Clearly there is an advantage in chrominance coding with respect to luminance slopes. Using only two segments the advantage is of the order of 0.2 bits/chrominance pel, whereas using four segments the advantage is of the order of 0.35 bits/chrominance pel.

5.2.4d Code Assignment

We mentioned earlier that the probability distribution of the quantizer output levels is highly nonuniform for intraframe as well as interframe predictive coders. This naturally lends itself to representation using code words of variable lengths (e.g., Huffman codes). The average bit-rate for such a code is usually very close to and is lower bounded by the entropy of the quantizer output signal as we saw in Chapter 3. A typical variable length code [5.2.23] for a previous pel DPCM coder with 16 quantizer levels is shown in Fig. 5.2.45. Inner levels occur much more often and, therefore, are represented by a smaller length word.

A Huffman code (see Chapter 3) requires a fairly reliable knowledge of the probability distribution of the quantized output. However, there are other procedures that work with approximate probabilities and still remain reasonably efficient even when the probabilities depart somewhat from their assumed values. Also in practice, it is found that the Huffman code is not too sensitive to changes in the probability distribution that take place for different pictures, and therefore, Huffman codes based on an average of many pictures remain quite efficient compared with codes designed on the basis of knowledge of the statistics of an individual picture. For most pictures, the entropy of the previous pel DPCM quantizer output (and consequently the average bit-rate using Huffman coding) is about one bit/pel less than the corresponding bit-rate for a fixed length code. Stated differently, variable length coding results in an improvement of about 6 dB in signal-to-noise ratio (SNR), with no change in the average transmission bit-rate. A similar conclusion can be arrived at quantitatively for a Laplacian probability distribution, (a good approximation for the differential signal). In this case, variable length coding can improve the SNR ratio by about 5.6 dB if the number of quantizer levels is large.

One of the problems with the use of variable length codes is that the output bit-rate from the source coder changes with local picture content. In order to transmit a variable bit-rate digital signal over a constant bit-rate channel, the source coder output has to be held temporarily in a first-in-first-out *buffer* that can accept input at a nonuniform rate, but whose output is conveyed to the channel at a uniform rate. Since in a practical system a finite size buffer must be used, any coder design must take into account the possibility of buffer overflow and underflow, and this depends in a complicated way upon the size of the buffer, the variable length code used, the image statistics, and the channel bit-rate. Buffer underflow is usually not a problem since PCM values could be inserted to increase the coder output whenever required. By using a channel

whose rate is somewhat higher than the entropy of the quantizer output, the probability of buffer overflow is markedly reduced. Also, by designing codes that minimize the probability of the length of the code words exceeding a certain number, the likelihood of buffer overflow can be reduced. However, since buffer overflow cannot always be prevented, strategies must be designed to gradually reduce the output bit-rate of the coder as the buffer begins to fill. This is particularly important in the case of interframe coders since the motion in a television scene is relatively bursty. Several techniques for reducing the bit-rate input to the buffer have been studied, e.g., reducing sampling rate, coarser quantization, etc. Such methods allow graceful degradation of picture quality when the source coder is overloaded, and will be discussed in more detail in Chapter 6.

5.3 Transform Coding of Blocks of Samples

With discrete transform coding, as we saw in Chapter 3, blocks of pels are transformed into another domain called the transform domain prior to coding and transmission. Transform coding has been found to have relatively good capability for bit-rate reduction, which comes about mainly from two mechanisms. First, not all of the transform domain coefficients need to be transmitted in order to maintain good image quality, and second, the coefficients that are coded need not be represented with full accuracy. Loosely speaking, transform coding is preferable to DPCM for compression to bit-rates below 2 or 3 bits per pel for single images. However, in those applications where complexity and cost are serious issues, low-pass filtering and reduction of pel sampling rate may be a preferable alternative.

5.3.1 Discrete Linear Orthonormal Transforms

With linear transforms each block of pels to be transformed is first arranged into a column vector $\boldsymbol{b}$ of length N, and with *orthonormal* (also known as unitary) linear transforms†

$$\boldsymbol{c} = \boldsymbol{T}\boldsymbol{b} \tag{5.3.1}$$

$$\boldsymbol{b} = \boldsymbol{T}'\boldsymbol{c} \tag{5.3.2}$$

where $\boldsymbol{T}$ is the $N \times N$ transform matrix, and $\boldsymbol{c}$ is the column vector of transform coefficients. If the mth column of matrix $\boldsymbol{T}'$ is denoted by $\boldsymbol{t}_m$ then‡

---

† Prime denotes conjugate transpose.

‡ $\delta_{mn} = 1$ if $m = n$, $\delta_{mn} = 0$ otherwise.

$$\boldsymbol{t}_m' \, \boldsymbol{t}_n = \delta_{mn} \tag{5.3.3}$$

which is where the term orthonormal arises. The vectors $\boldsymbol{t}_m$ are *orthonormal basis* vectors of the unitary transform $\boldsymbol{T}$, and using them Eq. (5.3.2) can be written as

$$\boldsymbol{b} = \sum_{m=1}^{N} c_m \boldsymbol{t}_m \tag{5.3.4}$$

where c_m is the mth element of $\boldsymbol{c}$, given by

$$c_m = \boldsymbol{t}_m' \, \boldsymbol{b} \tag{5.3.5}$$

The first basis vector $\boldsymbol{t}_1$ consists of constant values for practically all transforms of interest, i.e., it corresponds to zero spatial frequency and is given by

$$\boldsymbol{t}_1' = \frac{1}{\sqrt{N}} (1,1,1...1) \tag{5.3.6}$$

For this reason c_1 is often called the DC coefficient. Thus, if $b_{\max}$ is the largest possible pel value, then $\sqrt{N} \; b_{\max}$ is the largest possible value* for c_1.

The vector $\boldsymbol{b}$ could be constructed from N successive pels of a raster scan line, in which case the transform coding exploits one-dimensional horizontal correlation between pels. Alternatively, $\boldsymbol{b}$ could be constructed from a two-dimensional $L \times L$ array of image pels by simply laying the L-pel columns (or rows) end to end to form a single column vector of length $N = L^2$, in which case the transform coding exploits both horizontal and vertical correlation in the image.

A third possibility is to denote the $L \times L$ array of image pels by the square matrix $\boldsymbol{B} = [b_{ij}]$ and *separate* the transform into two steps. The first step is to transform the *rows* of $\boldsymbol{B}$ with a length L transform in order to exploit horizontal correlation in the image. Next, the *columns* of $\boldsymbol{B}$ are transformed in order to exploit vertical image correlation. The combined operation can be written

---

* Some discussions of transform coding in the literature use $c_i / \sqrt{N}$ instead of the c_i that we have defined, in order that the DC coefficient have the same dynamic range as the original pels.

$$c_{mn} = \sum_{i=1}^{L} \sum_{j=1}^{L} b_{ij} t_{nj} t_{mi} \tag{5.3.7}$$

where the t's are the elements of the $L \times L$ transform matrix $\boldsymbol{T}$ and $\boldsymbol{C} = [c_{mn}]$ is the $L \times L$ matrix of transform coefficients. Assuming the one dimensional basis vectors are indexed in order of increasing spatial frequency, then subscript m indexes vertical spatial frequency, and n indexes horizontal spatial frequency. If f_m and f_n are the vertical and horizontal spatial frequencies of the respective basis vectors, then the spatial frequency corresponding to coefficient c_{mn} is given by

$$f_{mn} = \sqrt{f_m^2 + f_n^2} \quad \frac{\text{cycles}}{\text{distance}} \tag{5.3.8}$$

For unitary transforms, the inverse operation is simply†

$$b_{ij} = \sum_{m=1}^{L} \sum_{n=1}^{L} c_{mn} t_{nj}^* t_{mi}^* \tag{5.3.9}$$

Matrices $\boldsymbol{C}$, $\boldsymbol{T}$ and $\boldsymbol{B}$ are related by

$$\boldsymbol{C} = \boldsymbol{T}\boldsymbol{B}\boldsymbol{T}^t \tag{5.3.10}$$

and for unitary $\boldsymbol{T}$ the inverse operation is simply

$$\boldsymbol{B} = \boldsymbol{T}'\boldsymbol{C}\boldsymbol{T}^* \tag{5.3.11}$$

The separable transformation requires $2L^3$ multiplications and additions, whereas the column transformation of Eq. (5.3.1) requires $N^2 = L^4$ such operations. Thus, the separable transform would normally be preferred for the usual case of $L > 2$. Extension of separable transforms to dimension higher than two is straightforward.

If $\{\boldsymbol{t}_m\}$ are the basis vectors of the Lth order transform $\boldsymbol{T}$, then Eq. (5.3.11) can be written

$$\boldsymbol{B} = \sum_{m=1}^{L} \sum_{n=1}^{L} c_{mn} \boldsymbol{t}_m^* \boldsymbol{t}_n' \tag{5.3.12}$$

† Asterisk denotes complex conjugate (which is ignored for real transforms).

That is, $\boldsymbol{B}$ can be thought of as a linear combination of $L \times L$ *basis images*

$$\{\boldsymbol{t}_m^* \boldsymbol{t}_n'\} \tag{5.3.13}$$

Eq. (5.3.12) is also known as an *outer product* expansion of $\boldsymbol{B}$.

We saw in Chapter 3 that linear transformation is an information preserving process. We also saw that a major objective of transform coding is to produce *statistically independent* (or at least uncorrelated) transform coefficients so that they can be coded independently of each other with good efficiency. Another objective is *energy compaction,* which means concentrating the energy in a few coefficients and thereby making as many coefficients as possible small enough that they need not be transmitted. Most of the unitary transforms discussed here achieve both objectives reasonably well.

We now define several unitary transforms of interest and discuss efficient methods for computing them.

5.3.1a Discrete Fourier Transform (DFT,RDFT)

The $N \times N$ unitary, symmetric Discrete Fourier Transform (DFT) matrix $\boldsymbol{T}$ has elements

$$t_{mi} = \frac{1}{\sqrt{N}} \exp\left[-\frac{2\pi}{N}\sqrt{-1}\,(i-1)(m-1)\right] \quad i,m = 1...N \tag{5.3.14}$$

For $N = 8$, matrix $\boldsymbol{T}$ is shown in Fig. 5.3.1. Since in our case $\boldsymbol{b}$ is real, the complex coefficients of $\boldsymbol{c}$ have conjugate symmetry, i.e.,

$$c_{2+p} = c_{N-p}' \qquad 0 \le p < N-2 \tag{5.3.15}$$

Therefore, reconstruction of the N pels of $\boldsymbol{b}$ requires only the transmission of at most N real values.

The unitary *Real* Discrete Fourier Transform (RDFT), having N real coefficients, can be derived by normalizing the real and imaginary parts of the DFT basis vectors as follows:

1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
1.000	.707	.000	-.707	-1.000	-.707	.000	.707
1.000	.000	-1.000	.000	1.000	.000	-1.000	.000
1.000	-.707	.000	.707	-1.000	.707	.000	-.707
1.000	-1.000	1.000	-1.000	1.000	-1.000	1.000	-1.000
1.000	-.707	.000	.707	-1.000	.707	.000	-.707
1.000	.000	-1.000	.000	1.000	.000	-1.000	.000
1.000	.707	.000	-.707	-1.000	-.707	.000	.707

.000	.000	.000	.000	.000	.000	.000	.000
.000	-.707	-1.000	-.707	.000	.707	1.000	.707
.000	-1.000	.000	1.000	.000	-1.000	.000	1.000
.000	-.707	1.000	-.707	.000	.707	-1.000	.707
.000	.000	.000	.000	.000	.000	.000	.000
.000	.707	-1.000	.707	.000	-.707	1.000	-.707
.000	1.000	.000	-1.000	.000	1.000	.000	-1.000
.000	.707	1.000	.707	.000	-.707	-1.000	-.707

Fig. 5.3.1 Transform matrix for the Discrete Fourier Transform (DFT) with $N = 8$. The top is the real part; the bottom is the imaginary part.

$$
\begin{aligned}
t_1^R &= t_1 \\
t_2^R &= \sqrt{2}\,\mathrm{Re}(t_2) \quad t_3^R = \sqrt{2}\,\mathrm{Im}(t_2) \\
t_4^R &= \sqrt{2}\,\mathrm{Re}(t_3) \quad t_5^R = \sqrt{2}\,\mathrm{Im}(t_3) \\
&\vdots \\
t_N^R &= \begin{cases} \sqrt{2}\,\mathrm{Im}\left[t_{\frac{N+1}{2}}\right], & N \text{ odd} \\ t_{\frac{N+2}{2}}, & N \text{ even} \end{cases}
\end{aligned}
\tag{5.3.16}
$$

It is instructive to plot real basis vectors as waveforms, since amplitudes, phases and frequencies may then be readily compared. The RDFT basis vectors for $N = 16$ are shown in Fig. 5.3.2.

The DFT basis vectors t_m are complex sinusoids. Thus, the DFT is a spectral decomposition with the energy of frequency $(m-1)/N$ being given by $|c_m|^2$. We saw in Chapter 3 that for most images the spectral energy generally decreases as a function of frequency.† Thus, the DFT results in a compaction of energy into the lower index transform coefficients.

A major feature of the DFT is the existence of well known fast algorithms, called Fast Fourier Transforms or FFT's.[5.3.1-2] Whereas a direct matrix multiplication DFT requires approximately N^2 complex multiplications and additions, FFT's require only about $N\log_2 N$ such operations if N is a power of two. This efficiency is achieved by exploiting the periodicities that exist within the Fourier transform matrix $\boldsymbol{T}$. FFT algorithms have also been devised for the case where N is not a power of two. Other advantages of FFT's in comparison with the direct DFT include reduced storage requirements and reduced round-off error. Most reasonably sized computer facilities contain one or more FFT implementations.*

---

† For $m \le (N+2)/2$. Components above this frequency are post-aliasing terms due to the sampling process.

* Many FFT implementations do not include the factor $1/\sqrt{N}$ in Eq. (5.3.14). Instead they include a factor $1/N$ in the inverse FFT. Also, a few of them compute the transform coefficients in the order $m = 2,3,\ldots,N,N+1$.

Since $\boldsymbol{b}$ is real, the Discrete Fourier transformation can be made more efficient by taking advantage of the conjugate symmetry of the complex coefficients. This is illustrated (for N even)[5.3.1] by the following procedure:

1) Define the complex vector $\boldsymbol{w}$, having length $N/2$, to be

$$\begin{aligned} \text{Re}\ (\boldsymbol{w}) &= (b_1 b_3 b_5 \ldots b_{N-1})' \\ \text{Im}\ (\boldsymbol{w}) &= (b_2 b_4 b_6 \ldots b_N)' \end{aligned} \tag{5.3.17}$$

2) Take the length $\frac{N}{2}$ DFT (or FFT‡) of $\boldsymbol{w}$ to obtain the complex coefficients $\{W_m,\ m = 1 \ldots \frac{N}{2}\}$.

3) For $2 \le m < \frac{N+5}{4}$ form the intermediate complex values:

$$X_m = \frac{1}{\sqrt{8}} \left[W_m + W'_{\frac{N}{2}+2-m} \right] \tag{5.3.18}$$

$$Y_m = \frac{\sqrt{-1}}{\sqrt{8}} \left[W_m - W'_{\frac{N}{2}+2-m} \right] \exp\left[- \frac{2\pi\sqrt{-1}\,(m-1)}{N} \right]$$

4) Then the complex coefficients for the DFT of $\boldsymbol{b}$ are given by:

‡ Also, multiply by $\sqrt{\frac{2}{N}}$ if not included in the FFT.

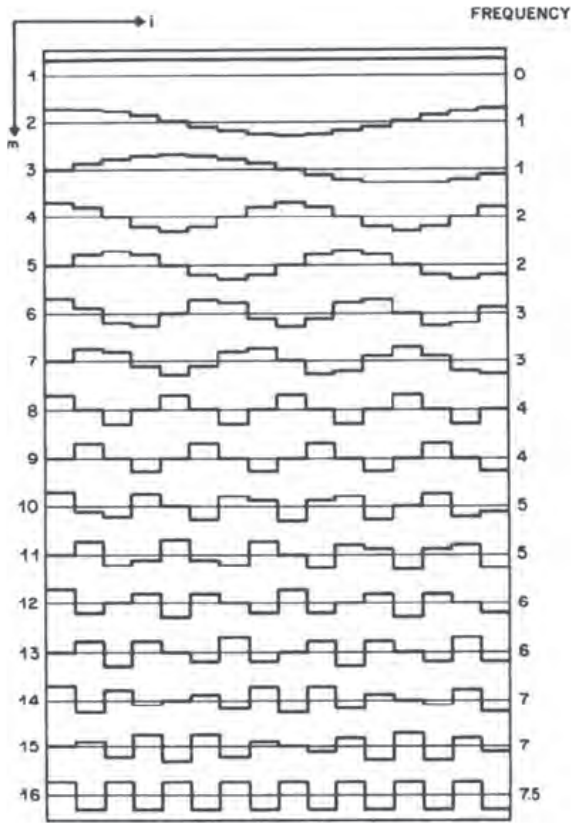


Fig. 5.3.2 Basis vectors for the Real Discrete Fourier Transform (RDFT) with $N = 16$. For each m the elements of t_m^R are plotted.

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

$$\frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Fig. 5.3.3 Transform matrices of the Walsh Hadamard Transform (WHT) with $N = 4$ and 8.

$$c_1 = \frac{1}{\sqrt{2}}\,[\mathrm{Re}(W_1)+\mathrm{Im}(W_1)]$$

$$\left\{c_m = X_m - Y_m\ ,\ 2 \le m < \frac{N+5}{4}\right\}$$

$$\left\{c_{\frac{N}{2}+2-m} = X'_m + Y'_m\ ,\ 2 \le m < \frac{N+5}{4}\right\} \qquad (5.3.19)$$

$$c_{\frac{N}{2}+1} = \frac{1}{\sqrt{2}}\,[\mathrm{Re}(W_1)-\mathrm{Im}(W_1)]$$

$$\left\{c_{N-p} = c'_{2+p}\ ,\ 0 \le p \le \frac{N}{2} - 2\right\}$$

For coding purposes, only N real numbers need be computed. In particular, the coefficients of the unitary Real Discrete Fourier Transform (RDFT) are given (for N even) by:

$$\begin{aligned} c_1^R &= c_1 \\ c_2^R = \sqrt{2}\,\mathrm{Re}(c_2) \quad & c_3^R = \sqrt{2}\,\mathrm{Im}(c_2) \\ c_4^R = \sqrt{2}\,\mathrm{Re}(c_3) \quad & c_5^R = \sqrt{2}\,\mathrm{Im}(c_3) \\ &\vdots \\ c_N^R &= c_{\frac{N}{2}+1} \end{aligned} \qquad (5.3.20)$$

In order to perform the inverse DFT (or inverse RDFT) we simply reverse the above procedure. That is, we solve for X_m and Y_m, then solve for W_m, then perform the inverse DFT (or inverse FFT) and finally reorder $\boldsymbol{w}$ to obtain $\boldsymbol{b}$.

5.3.1b Walsh-Hadamard Transform (WHT)

The WHT is most easily described recursively. Let $\boldsymbol{H}_1 = 1$, and for $N = 2^n$ define

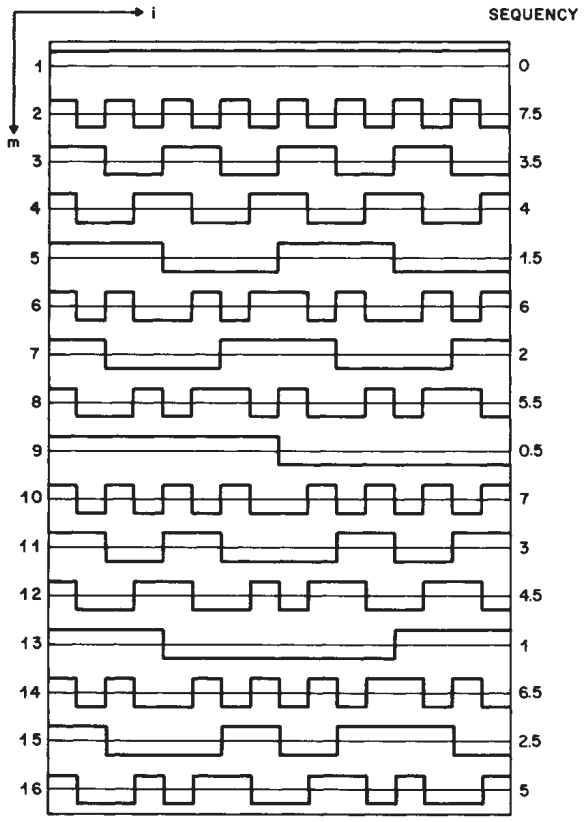


Fig. 5.3.4 Basis vectors for the WHT with $N = 16$. For each vector the sequency is the number of zero crossings divided by two.

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

$$\frac{1}{\sqrt{8}}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

Fig. 5.3.5 Sequency ordered transform matrices for the WHT with $N = 4$ and 8.

$$\boldsymbol{H}_{2N} = \begin{bmatrix} \boldsymbol{H}_N & \boldsymbol{H}_N \\ \boldsymbol{H}_N & -\boldsymbol{H}_N \end{bmatrix} \tag{5.3.21}$$

Then the unitary, symmetric WH transform matrix is given by

$$\boldsymbol{T} = \frac{1}{\sqrt{N}} \boldsymbol{H}_N = \boldsymbol{T}' \tag{5.3.22}$$

For $N = 4$ and 8, matrices $\boldsymbol{T}$ are shown in Fig. 5.3.3 for the WHT. The main advantage of the WHT is that apart from the factor $1/\sqrt{N}$, the computation requires only addition and subtraction, as opposed to most other transforms that require multiplication as well. Moreover, a fast algorithm exists that requires only about $N \log_2 N$ operations.[5.3.4]

An alternative definition also exists that is often more useful for hardware implementation or computer calculation. For this representation we require the binary coefficients of integers less than N. We again assume $N = 2^n$, and for $k < N$ we write

$$\begin{aligned} k &= k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \ldots + k_1 2^1 + k_0 2^0 \\ &= \sum_{s=0}^{n-1} k_s 2^s \quad \text{where} \quad k_s = 0 \text{ or } 1 \ . \end{aligned} \tag{5.3.23}$$

Next, for integers $k, \ell < N$ we introduce the notation

$$\langle k, \ell \rangle \triangleq \sum_{s=0}^{n-1} k_s \ell_s \tag{5.3.24}$$

Then it can be shown that the WH transform matrix T has elements given by

$$t_{mi} = \frac{1}{\sqrt{N}} (-1)^{\langle m-1, i-1 \rangle} \quad m,i = 1 \ldots N \tag{5.3.25}$$

The concept of frequency can also be extended to the WHT, in which case it is called *sequency*. The sequency of a WHT basis vector $\boldsymbol{t}_m$ is defined as the number of sign changes within it divided by two. If the basis vectors $\boldsymbol{t}_m$ are plotted as 2-level waveforms, as in Fig. 5.3.4, then the sequency is half the number of zero crossings, which corresponds to the definition of frequency by sinusoids. As with the DFT, the energy in the WHT coefficients for images tends to decrease with sequency.

Unfortunately, the WHT definitions of Eqs. (5.3.22) and (5.3.25) do not have their basis vectors ordered according to sequency. To do so requires the following notation for $k,\ell < N$. Referring to Eq. (5.3.23)

$$\bar{k}_0 = k_{n-1} \tag{5.3.26}$$

$$\bar{k}_s = k_{n-s} + k_{n-s-1} \qquad s = 1 \ldots n-1$$

$$\langle \bar{k}, \ell \rangle = \sum_{s=0}^{n-1} \bar{k}_s \ell_s \tag{5.3.27}$$

Then it can be shown that the *sequency ordered* WH transform matrix $\boldsymbol{T}$ has elements given by

$$t_{mi} = \frac{1}{\sqrt{N}} (-1)^{\langle \overline{m-1}, i-1 \rangle} \tag{5.3.28}$$

$$m,i = 1 \ldots N$$

Sequency ordered WHT matrices are shown in Fig. 5.3.5 and basis vectors for $N = 16$ in Fig. 5.3.6. Note that T is still unitary and symmetric. A fast algorithm also exists for the sequency ordered WHT.[5.3.2]

WHT transform matrices when N is not a power of two also exist, although not for every value of N.[5.3.4] However, for image coding purposes $N = 2^n$ is almost universally used.

5.3.1c Karhunen-Loeve Transform (KLT)

The *KL* transform has coefficients that are uncorrelated. Its matrix $\boldsymbol{T}$ is derived from the $N \times N$ correlation matrix of the image pel blocks $\boldsymbol{b}$ that have been biased to have zero mean. That is,

$$\boldsymbol{R} \triangleq E[(\boldsymbol{b} - E\boldsymbol{b})(\boldsymbol{b} - E\boldsymbol{b})'] \tag{5.3.29}$$

or in terms of the elements of $\boldsymbol{R}$ and $\boldsymbol{b}$

$$r(i,j) = E[(b_i - Eb_i)(b_j - Eb_j)] \tag{5.3.30}$$

A slightly suboptimum procedure is to define the sample mean of each vector

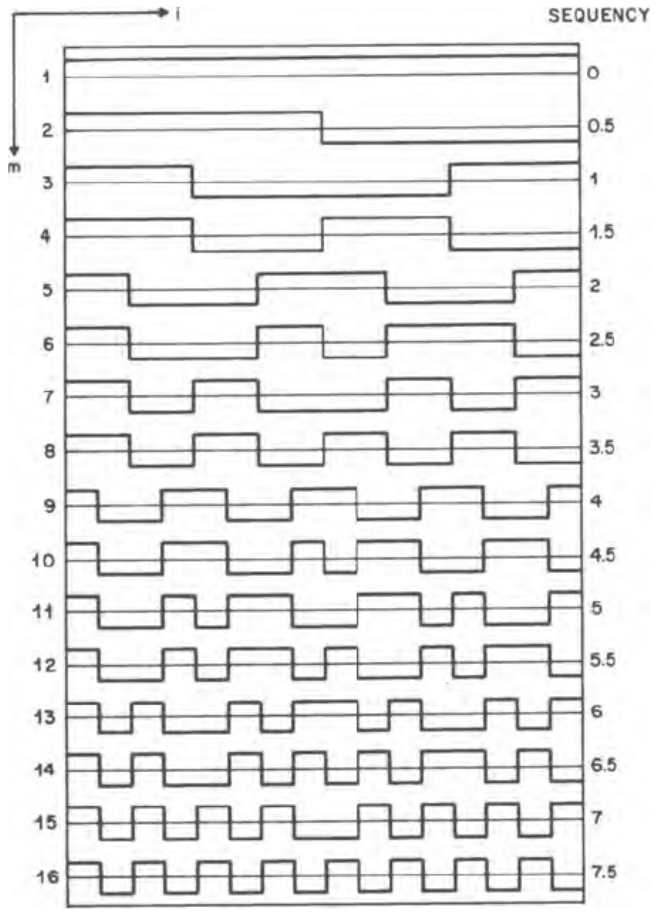


Fig. 5.3.6 Sequency ordered basis vectors for the WHT with $N = 16$.

Fig. 5.3.7 Images used for coding and statistics. (a) "Karen" has much more stationary statistics than (b) "Stripes."

$$\mu = \frac{1}{N} \sum_{i=1}^{N} Eb_i \tag{5.3.31}$$

and use the matrix†

$$r(i,j) = E[(b_i - \mu)(b_j - \mu)] \tag{5.3.32}$$

which is also positive definite and a legitimate correlation matrix. In fact, setting $\mu = 0$ in Eq. (5.3.32) does not usually cause significant penalty when coding images. In any event, the KLT basis vectors are the real, orthonormalized eigenvectors of $\boldsymbol{R}$, i.e.,

$$\begin{aligned} \boldsymbol{R}\boldsymbol{t}_m &= \lambda_m \boldsymbol{t}_m \quad , \\ \boldsymbol{t}_m' \boldsymbol{t}_n &= \delta_{mn} \end{aligned} \tag{5.3.33}$$

where the eigenvalues $\{\lambda_m\}$ are nonnegative.

For example, Fig. 5.3.8 shows KLT eigenvectors for the picture "Karen" in Fig. 5.3.7a using one-dimensional blocks of pels with $N = 16$ and $\mu = 0$. The eigenvectors are arranged according to decreasing eigenvalue. Note that, for the most part, this also corresponds to ordering according to increasing frequency, i.e., number of zero crossings in the basis vector. Fig. 5.3.9 shows similar data for the first order Markov correlation model defined in Chapter 3. In this case, statistics are stationary with pel variance σ^2 and adjacent pel correlation $0 < \rho < 1$. The correlation function is given by

$$r(i,j) = \sigma^2 \rho^{|i-j|} \tag{5.3.34}$$

In the figure we assume a value of adjacent pel correlation $\rho = 0.95$. Note the similarity between the eigenvectors of Fig. 5.3.8 and Fig. 5.3.9. Such similarity is generally seen for images such as "Karen" that have "nearly stationary" statistics.

We now consider larger blocks of pels, both one dimensional and two dimensional. Fig. 5.3.10 shows the first few eigenvectors for the picture "Karen" using one-dimensional blocks of pels with $N = 64$. Fig. 5.3.11 shows results when the $N = 64$ blocks are made from $L \times L$ two-dimensional blocks of pels (here $L = 8$). Note that for the two-dimensional blocks many periodicities of length L are apparent. This reflects the vertical correlation between pels in successive scan lines, which occurs in most images.

† Note that in general $\mu \neq Eb_i \neq Eb_j$.

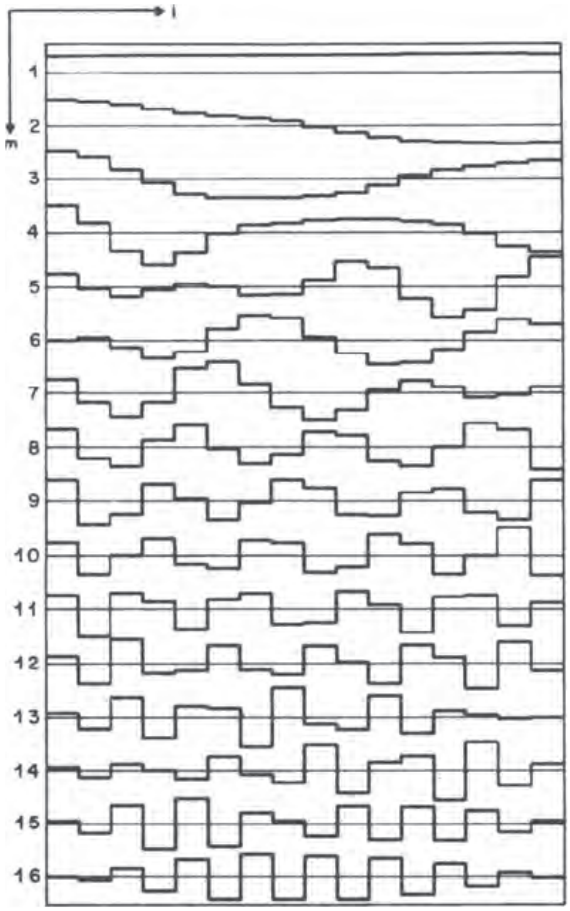


Fig. 5.3.8 KLT basis vectors for the image "Karen" using one-dimensional blocks with $N = 16$ and $\mu = 0$. For each m the elements of t_m are plotted.

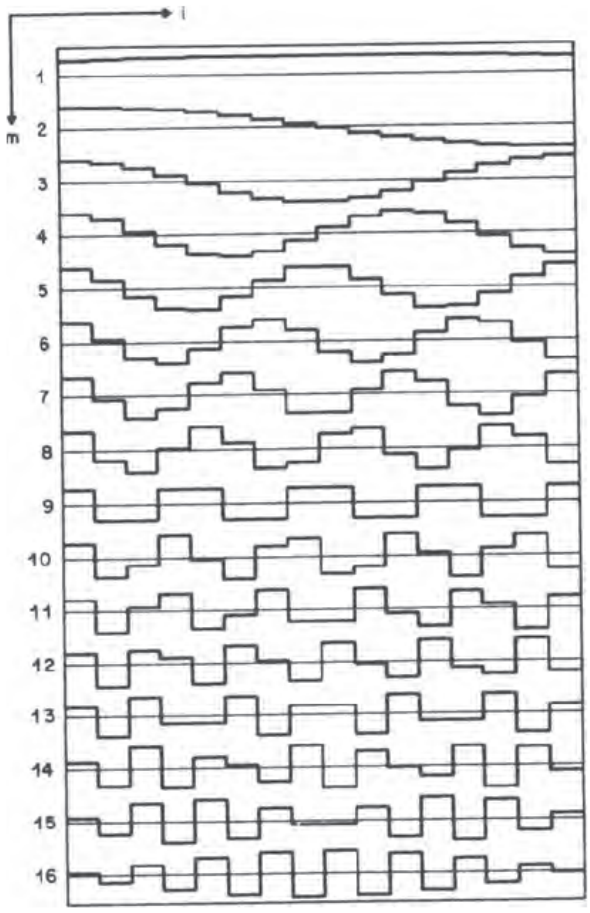


Fig. 5.3.9 KLT basis vectors ($1D$, $N=16$) for the first order Markov correlation model with adjacent pel correlation $\rho=0.95$.

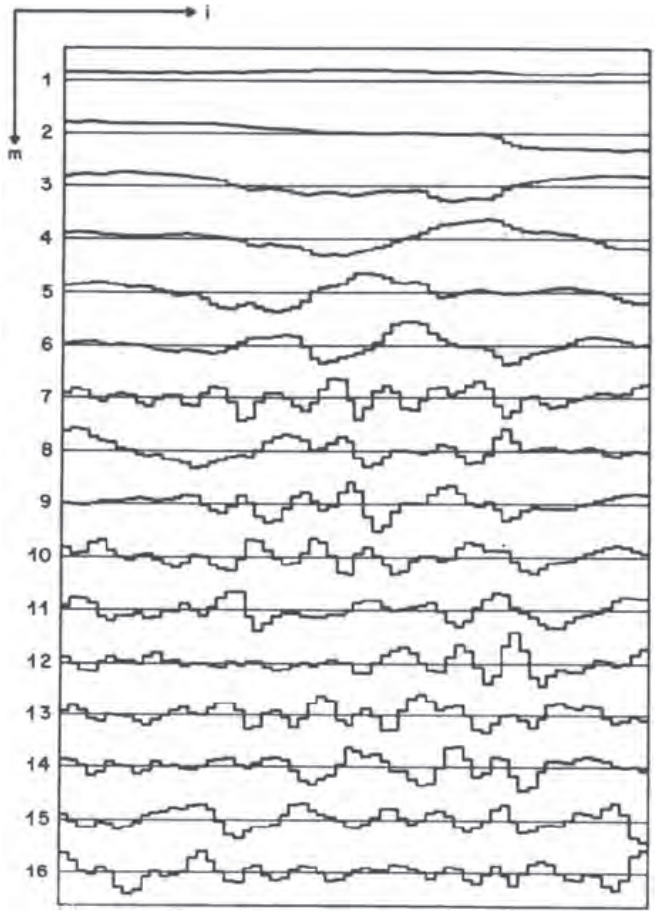


Fig. 5.3.10 First 16 KLT basis vectors for the image "Karen" using one-dimensional, $N = 64$ blocks.

Fig. 5.3.12 shows one-dimensional, $N = 64$ eigenvectors for the picture "Stripes," which has highly nonstationary statistics. Note that these eigenvectors attempt to match the horizontal periodicities due to the stripes in the image, and are very different compared to the basis vectors of "Karen" shown in Fig. 5.3.10.

Once the KLT basis vectors have been determined, as we saw in Chapter 3, the transformation is then given by

$$\boldsymbol{c} = \boldsymbol{T}\boldsymbol{b} \tag{5.3.35}$$

and

$$E(c_m c_n) = \lambda_m \delta_{mn} \tag{5.3.36}$$

That is, the *KL* transform coefficients are uncorrelated, and their mean square values (MSV's) are equal to their respective eigenvalues. Thus, if the KLT basis vectors are ordered according to decreasing eigenvalues, i.e., λ_1 largest and λ_N smallest, then a compaction of energy into the lower index transform coefficients will result. It was shown in Chapter 3 that with the mean square error distortion criterion, the KLT achieves the best energy compaction of any linear transform.

A drawback of the KLT may occur with the coding of two dimensional $L \times L$ blocks of pels $\boldsymbol{B}$. If the L columns are laid end to end to make a length $N = L^2$ column, then a nonseparable KLT of order L^2 is required. If the separable two-dimensional transform on Eq. (5.3.10) is used, then a KLT only of order L is needed. However, if the image statistics are highly nonstationary then the performance of the separable transform may suffer.

5.3.1d Discrete Cosine Transform (DCT)

In Chapter 3 we defined the Discrete Cosine Transform or DCT.[5.3.6] In recent years, the DCT has become the most widely used of the unitary transforms, and under certain circumstances, as we shall see later, its performance can come close to that of the KLT. The elements of the DCT transform matrix $\boldsymbol{T}$ are given by [$\delta_0 = 1, \delta_p = 0$, for $p > 0$]

$$t_{mi} = \sqrt{\frac{2-\delta_{m-1}}{N}} \cos\left\{\frac{\pi}{N}\left[i-\frac{1}{2}\right](m-1)\right\} \tag{5.3.37}$$

$$i,m = 1 \ldots N$$

Thus, DCT basis vectors $\boldsymbol{t}_m$ are sinusoids with frequency indexed by m.

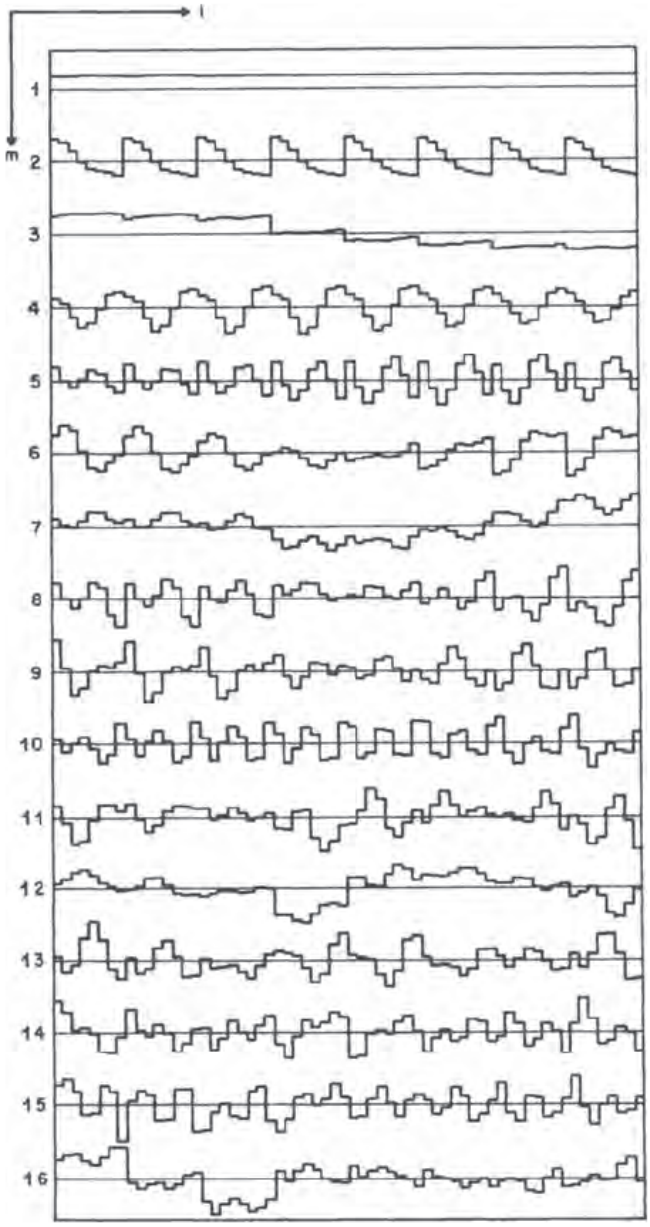


Fig. 5.3.11 First 16 KLT basis vectors for the image "Karen" using two-dimensional, nonseparable 8×8 blocks.

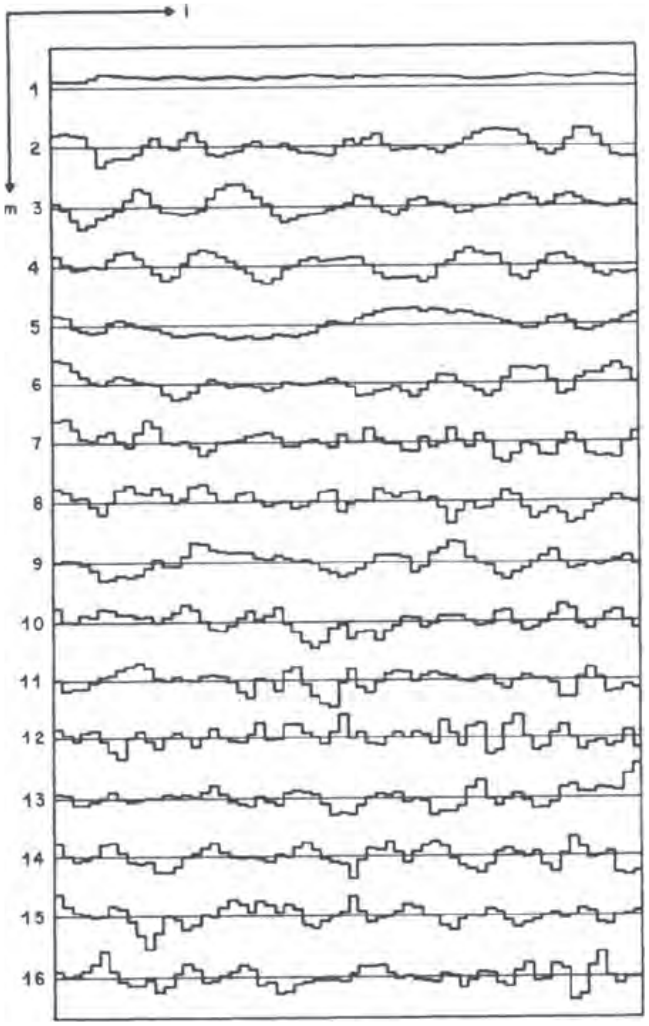


Fig. 5.3.12 First 16 KLT basis vectors for the image "Stripes" using one-dimensional $N = 64$ blocks.

For $N = 16$ the DCT basis vectors are shown in Fig. 5.3.13. Note the similarity between the DCT basis vectors and the KLT basis vectors of Fig. 5.3.8 and Fig. 5.3.9. This experimentally observed property is, in part, responsible for the generally good performance of the DCT compared with other transforms, as we shall see later.

Computation of the DCT is greatly facilitated by observing that the elements of $\boldsymbol{T}$ in Eq. (5.3.37) can be written as the real part of

$$\begin{aligned}
&\frac{\sqrt{2(2-\delta_{m-1})}}{\sqrt{2N}} \quad \exp\left[\frac{2\pi\sqrt{-1}}{2N}\left(i-1+\frac{1}{2}\right)(m-1)\right] \\
&\qquad = \sqrt{2(2-\delta_{m-1})}\ exp\left[\frac{\pi\sqrt{-1}}{2N}(m-1)\right] \\
&\qquad \times \frac{1}{\sqrt{2N}}\ exp\left[\frac{2\pi\sqrt{-1}}{2N}(i-1)(m-1)\right] \qquad (5.3.38) \\
&\qquad \triangleq F_m \times F_{mi}
\end{aligned}$$

The second factor F_{mi} is recognized as element (m,i) of the $2N$th order DFT matrix. Thus, the DCT can be efficiently computed as follows:[5.3.6]

1) Take the FFT* of the length $2N$ vector

$$\boldsymbol{b}^+ \triangleq (b_1 b_2 ... b_N 000...0)' \qquad (5.3.39)$$

2) The first N of the resulting Fourier coefficients (indexed by m) are then multiplied by their respective F_m of Eq. (5.3.38).

3) Finally, the real part is taken to give the N DCT coefficients.

---

* Also, multiply by $\sqrt{\frac{2}{N}}$ if not included in the FFT.

The inverse DCT can be computed similarly since the elements t_{im} of the inversion matrix are also given by the real part of Eq. (5.3.38). The inverse procedure starts by constructing the length $2N$ vector of DCT coefficients

$$c^{+} = (c_1 c_2 ... c_N 00...0)' \tag{5.3.40}$$

Then each coefficient is multiplied by its respective F_m of Eq. (5.3.38) and the $2N$th order FFT taken. The real parts of the resulting first N values are the desired data.

An alternative and even faster DCT algorithm[5.3.7] is based on reordering the block of pels as follows:

$$\mathbf{v} \triangleq (b_1 b_3 b_5 b_7 ... b_8 b_6 b_4 b_2)' \tag{5.3.41}$$

that is odd pels first, followed by even pels in reverse order. Thus,

$$v_i = \begin{cases} b_{2i-1} & 1 \le i < \dfrac{N+1.5}{2} \\ b_{2N-2i+2} & \dfrac{N+1.5}{2} < i \le N \end{cases} \tag{5.3.42}$$

The term $N+1.5$ allows one definition for both even and odd N. The DCT coefficients can then be written

$$\begin{aligned} c_m &= \sqrt{\frac{2-\delta_{m-1}}{N}}\ \mathrm{Re} \sum_{i=1}^{N} b_i \exp\left[-\frac{\pi}{N}\sqrt{-1}\,(i-\frac{1}{2})(m-1)\right] \\ &= \sqrt{\frac{2-\delta_{m-1}}{N}}\ \mathrm{Re}\left[\sum_{i\ \mathrm{odd}} \cdots + \sum_{i\ \mathrm{even}} \cdots\right] \end{aligned} \tag{5.3.43}$$

Replacing i by $2i - 1$ and $2N - 2i+2$ in the two summands, respectively, we get

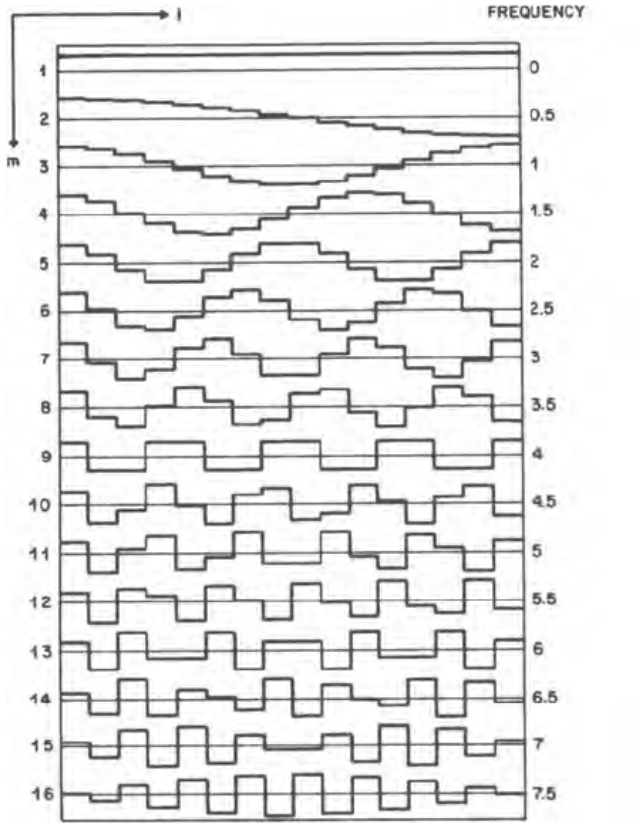


Fig. 5.3.13 Basis vectors for the DCT with $N = 16$.

N	Number of Computations For Matrix DCT	Fast DCT	Speedup Factor
4	16	8	2
8	64	24	2.7
16	256	64	4
32	1024	160	6.4
64	4096	384	10.7
138	16384	896	18.3
256	65536	2048	32

Fig. 5.3.14 Computational advantage of using the Fast DCT Algorithm compared with simple matrix multiplication.

$$c_m = \sqrt{\frac{2-\delta_{m-1}}{N}}\ \mathrm{Re}\sum_{i=1}^{N} v_i \exp\left[-\frac{2\pi}{N}\sqrt{-1}\left[i-\frac{3}{4}\right](m-1)\right]$$

$$= \sqrt{2-\delta_{m-1}}\ \mathrm{Re}\ (G_m V_m) \tag{5.3.44}$$

where

$$G_m = \exp\left[-\frac{\pi}{2N}\sqrt{-1}\ (m-1)\right] \tag{5.3.45}$$

and $\{V_m\}$ is the Nth order DFT of the reordered block of pels $\mathbf{v}$. Also, since $V_{N-p} = V'_{2+p}$ and $G_{N-p} = -\sqrt{-1}\ G'_{2+p}$, we have

$$\mathrm{Re}(G_{N-p}V_{N-p}) = -\mathrm{Im}(G_{2+p}V_{2+p}) \tag{5.3.46}$$

and, as with the DFT (for N even), the DCT can be performed using only an $\frac{N}{2}$-point FFT. Thus we have (for N even):

A FAST DCT ALGORITHM

1) Reorder the pels to form the complex vector $\boldsymbol{w}$, having length $N/2$, as follows:

$$\mathrm{Re}(\boldsymbol{w}) = (b_1 b_5 b_9 \ldots b_{12} b_8 b_4)'$$
$$\mathrm{Im}(\boldsymbol{w}) = (b_3 b_7 b_{11} \ldots b_{10} b_6 b_2)' \tag{5.3.47}$$

2) Take the DFT (or FFT†) of $\boldsymbol{w}$ to obtain the complex coefficients $\{W_m,\ m = 1 \ldots N/2\}$, form the complex values X_m and Y_m from Eqs. (5.3.18), and compute the complex DFT coefficients $\{V_m,\ 1 \le m \le 1 + N/2\}$ using Eqs. (5.3.19).

† Also, multiply by $\sqrt{\frac{2}{N}}$ if not included in the FFT.

3) Form the complex products $\{U_m = G_m V_m,\ 2 \le m \le N/2\}$.

4) The DCT coefficients are then given by

$$\begin{aligned} c_1 &= V_1 \\ c_m &= \sqrt{2}\ \mathrm{Re}(U_m), \qquad 2 \le m \le \frac{N}{2} \\ c_{\frac{N}{2}+1} &= V_{\frac{N}{2}+1} \\ c_{N-p} &= -\sqrt{2}\ \mathrm{Im}(U_{2+p}), \qquad 0 \le p \le \frac{N}{2}-2 \end{aligned} \tag{5.3.48}$$

The inverse DCT is the reverse of the above procedure. That is, we solve for U_m, then V_m, then X_m and Y_m, then W_m. An inverse $N/2$-point DFT (or inverse FFT) then gives $\boldsymbol{w}$, and a reordering produces $\boldsymbol{b}$. The computation time for the Fast DCT's increases as $N\log_2 N$, as opposed to the N^2 increase for direct matrix multiplication. For sizable N the computational advantage of using the Fast DCT Algorithm can be enormous, as shown in Fig. 5.3.14. For smaller N, other approaches may be more efficient. For example, the Appendix gives an implementation for block size 8 that uses *rotations*. See [5.3.7a] for details.

An indication of the compaction capabilities of the DCT can be obtained by examining the spectral energy distribution in the transform domain. Fig. 5.3.15 shows DCT coefficient mean square values (MSV) for the image "Karen", for various 1D and 2D block sizes, arranged according to increasing spatial frequency [see Eq. (5.3.8)] as well as arranged according to decreasing MSV. Note the roughly linear behavior of MSV's on a log-log scale for the separable 2D-DCT shown in Fig. 5.3.15c.

Results can also be derived using the mathematical models defined in Chapter 3. For example, with (real) separable 2D-transforms of $L \times L$ pel blocks $\boldsymbol{B} = [b_{ij}]$, we have from Eq. (5.3.7)

$$c_{mn} = \sum_{i,j=1}^{L} b_{ij} t_{mi} t_{nj} \tag{5.3.49}$$

From this, coefficient MSV's become

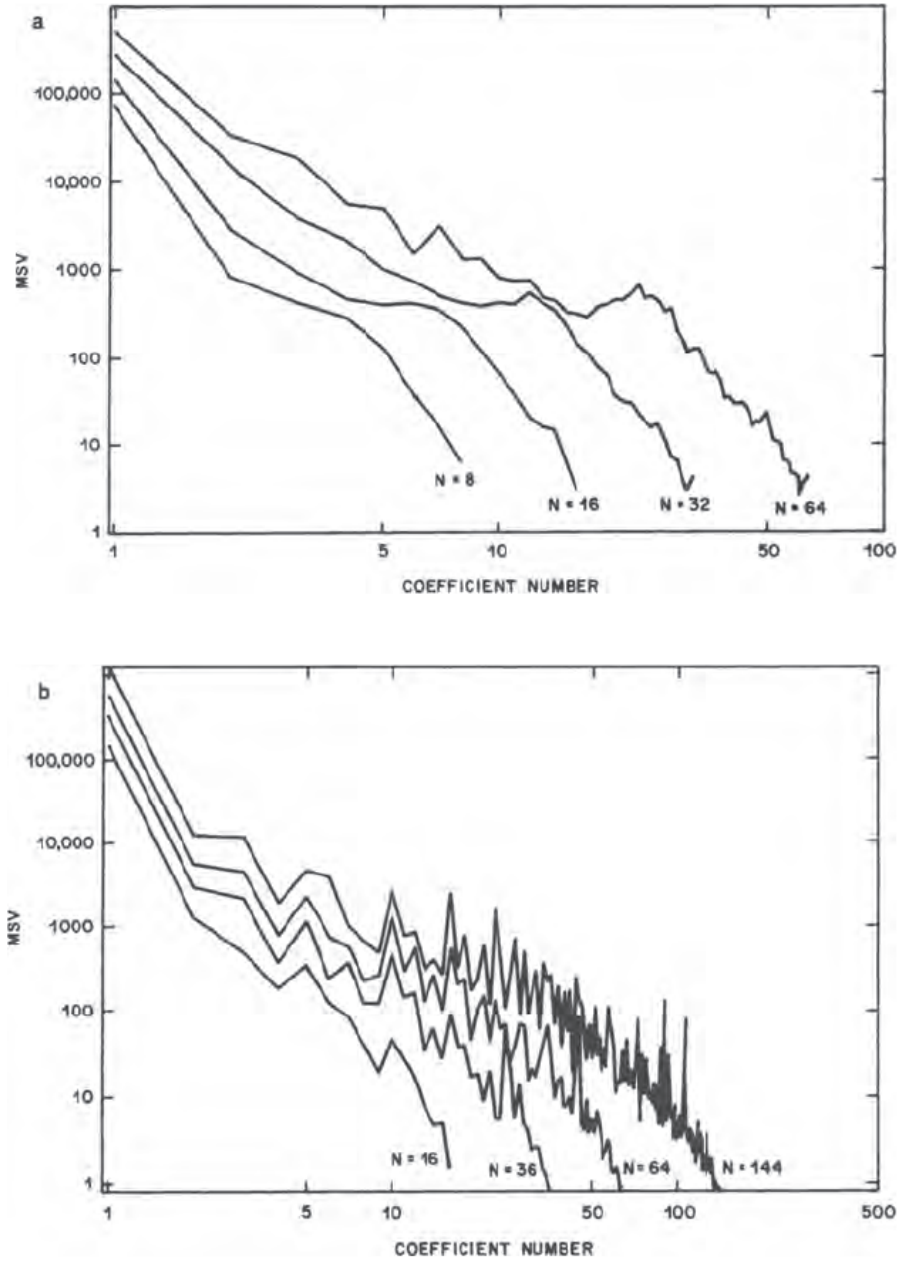


Fig. 5.3.15 DCT coefficient mean square values for the image "Karen" with various block sizes. (a) One-dimensional, arranged according to increasing frequency. (b) Separable two-dimensional, arranged according to increasing spatial frequency.

$$\sigma_{mn}^2 \triangleq E(c_{mn}^2) = E \sum_{i,j,k,\ell=1}^{L} b_{ij} b_{k\ell} t_{mi} t_{nj} t_{mk} t_{n\ell}$$

(5.3.50)

$$= \sum_{i,j,k,\ell=1}^{L} r_{ijk\ell} t_{mi} t_{nj} t_{mk} t_{n\ell}$$

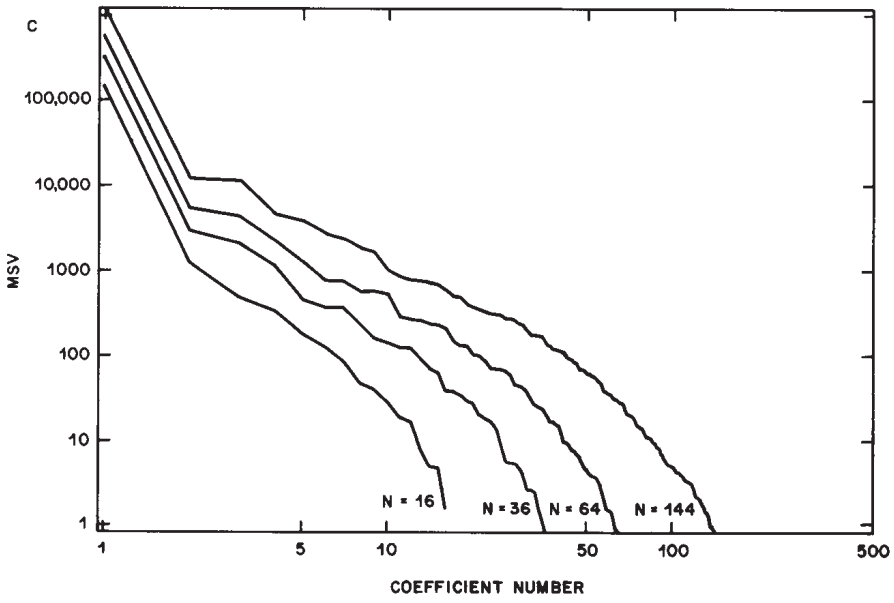


Fig. 5.3.15c Separable 2D arranged according to decreasing mean square value (MSV).

where the correlation $r_{ijk\ell} \triangleq E(b_{ij}b_{k\ell})$. For example, using an isotropic model with 8-bit PCM original data having range 0–255 and $MSV = 255^2/3$, the correlation function becomes

$$r_{ijk\ell} = \frac{255^2}{3}\,\rho^{\sqrt{(i-k)^2 + (j-\ell)^2}} \tag{5.3.51}$$

Fig. 5.3.16 shows DCT coefficient MSV's for the isotropic model with $\rho = 0.95$, for various 1D and 2D block sizes, again arranged according to increasing spatial frequency. Note again the linear behavior for the separable 2D-DCT.

5.3.1e Comparison of RDFT, WHT, KLT, DCT

So far, we have dwelt mainly on the computational aspects of discrete orthogonal transforms. We now compare the energy compaction performance of the four most often used transforms, namely the RDFT, WHT, KLT, and DCT. We have seen that unitary transformation preserves image energy or MSV, i.e.,

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N} E(b_i^2) = \frac{1}{N}\sum_{m=1}^{N} E(c_m^2) \tag{5.3.52}$$

Also, if only a certain fixed fraction $p < 1$ of the transform coefficients are transmitted in each block and used to form the reconstructed pels $\tilde{\boldsymbol{b}}$ then the mean square error becomes

$$\begin{aligned} MSE &= \frac{1}{N}\sum_{i=1}^{N} E\left[(b_i - \tilde{b}_i)^2\right] \\ &= \frac{1}{N}\sum_{m=1}^{N} E\left[(c_m - \tilde{c}_m)^2\right] \end{aligned} \tag{5.3.53}$$

Note that this does not include quantization error, which we will treat in the next section. If the coefficients are reindexed so that the fixed set of transmitted coefficients occur first, then

$$\tilde{c}_m = \begin{cases} c_m, & 1 \le m \le pN \\ 0, & m > pN \end{cases} \tag{5.3.54}$$

and

$$MSE = \frac{1}{N}\sum_{m>pN}^{N} E\left[c_m^2\right]$$

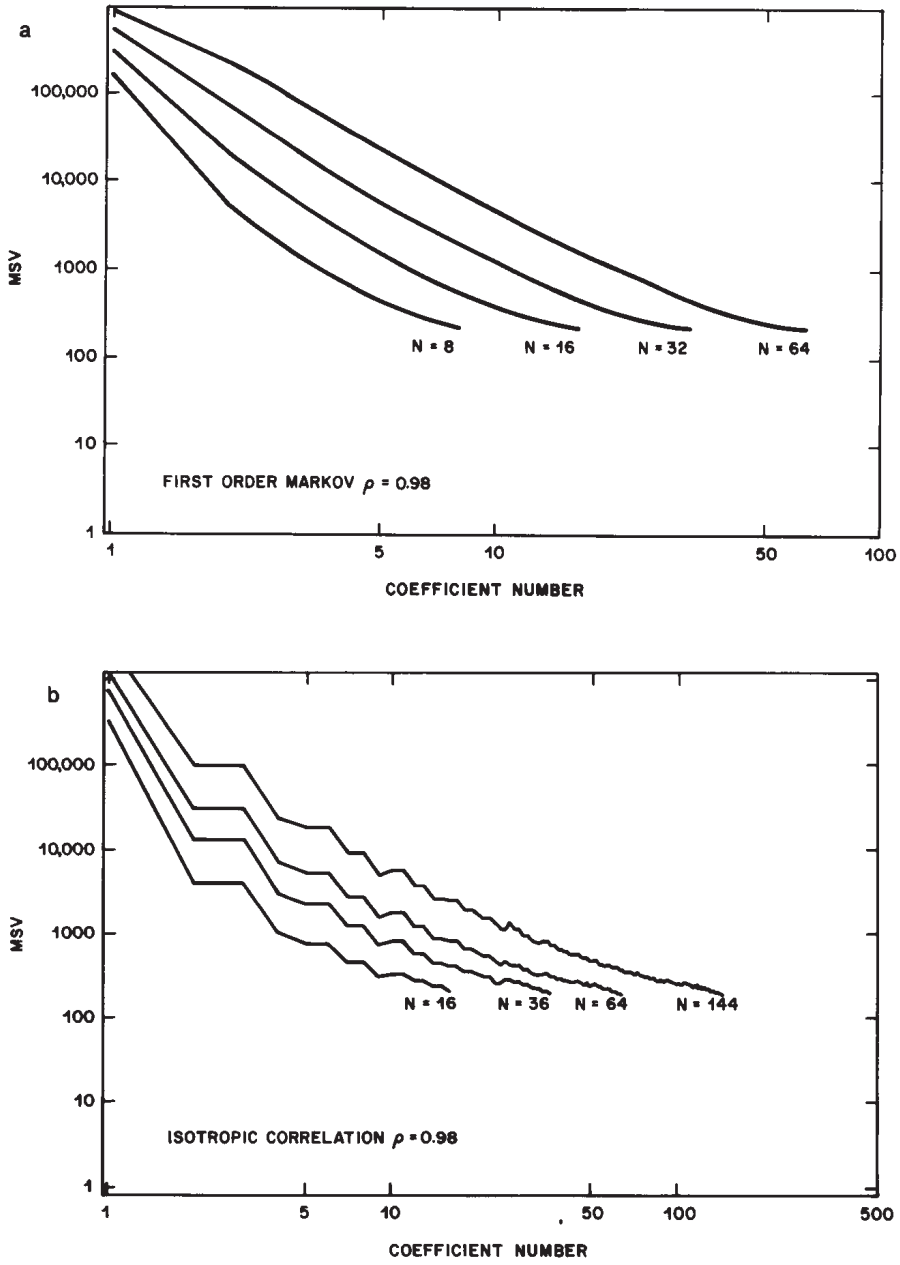


Fig. 5.3.16 DCT coefficients MSV's using the isotropic correlation model arranged according to increasing frequency ($\rho = 0.98$). (a) One-dimensional blocks. (b) Separable two-dimensional blocks.

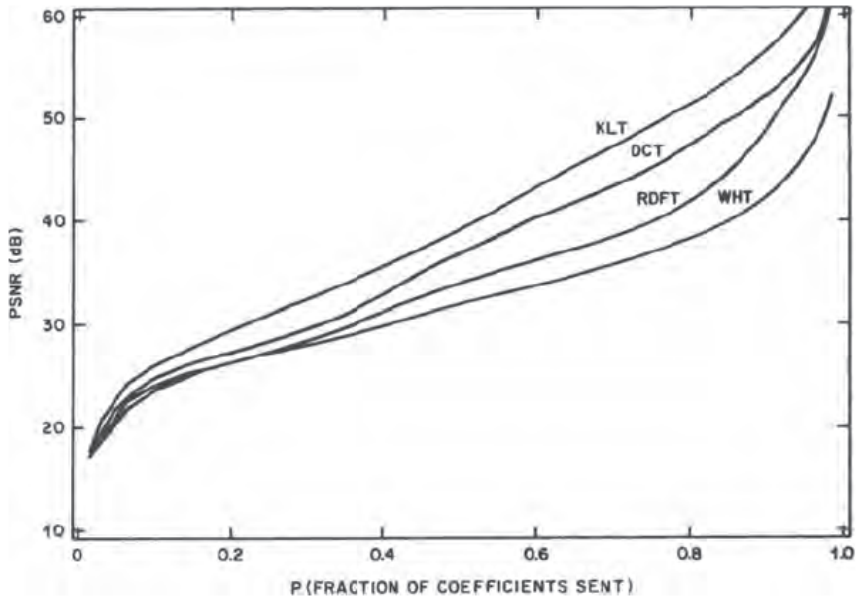


Fig. 5.3.17 Comparison of truncation errors for four different transforms. One-dimensional blocks, $N = 64$, were used with the image "Karen". Peak signal to rms noise ratio (PSNR) is plotted versus the fraction of coefficients sent.

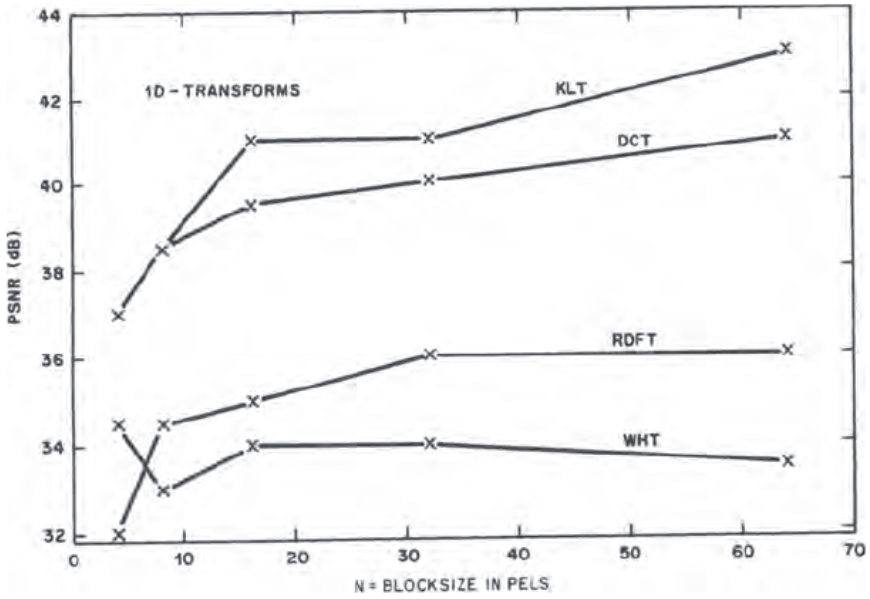


Fig. 5.3.18 Truncation Peak Signal to rms Noise ratio (PSNR) versus one-dimensional block size for the image "Karen" using four different transforms and keeping 60 percent of the coefficients ($p = 0.6$).

For 8-bit PCM original data, the range of pel values is 0-255. The peak-signal to rms noise ratio then becomes (in dB)

$$PSNR = 10 \log \left[\frac{255^2}{MSE} \right] dB \tag{5.3.55}$$

Although unweighted PSNR (see Section 4.3.2) in this context has some value for comparison purposes, we should not pay too much attention to the actual PSNR values themselves. In the first place the impairment is not subjectively weighted. In addition, the distortion is in fact a low-pass filtering that appears to the viewer as blurring. Such impairment is subjectively not well parameterized by PSNR.

Fig. 5.3.17 shows PSNR results for the four transforms applied to "Karen" (1D-64 pel blocks) when only pN ($p < 1$) of the transform coefficients are transmitted. In each case the coefficients having the largest MSV's are transmitted, a technique called *zonal sampling*.* We see that the KLT is the best performing transform in terms of energy compaction, followed by the DCT, RDFT and WHT, respectively. For other block sizes Fig. 5.3.18 shows PSNR results for $p = 0.6$, i.e., 60 percent of the coefficients are transmitted. The energy compaction generally improves as block size increases, and the relative performance of the transforms generally remains the same. However, the improvement is small for block sizes larger than about 16 pels.

We now consider the case where $\boldsymbol{b}$ is constructed from a two-dimensional $L \times L$ block of pels by simply laying the L-pel rows end-to-end. The resulting vector of $N = L^2$ pels has not only high adjacent pel correlation, but also high correlation between pels with separation L. Thus for example, RDFT coefficients are relatively large not only at low frequencies, but also at frequencies $1/L$, $2/L$, etc. Fig. 5.3.19 shows PSNR results when two-dimensional blocks are transform coded in this manner and only pN of the coefficients having the largest MSV are transmitted. The curves are generally higher than those of Fig. 5.3.17 by several dB.

With two-dimensional $L \times L$ blocks of pels the most often used approach is to separate the transformation using two L-dimensional transforms as described earlier in this chapter. Recall that with separable transforms we use an $L \times L$ block of pels $\boldsymbol{B}$ and transform the rows and columns separately via

---

* Note that this is not equivalent to sending the pN coefficients of each block having the largest magnitude. More on this subject appears in Section 5.3.3.

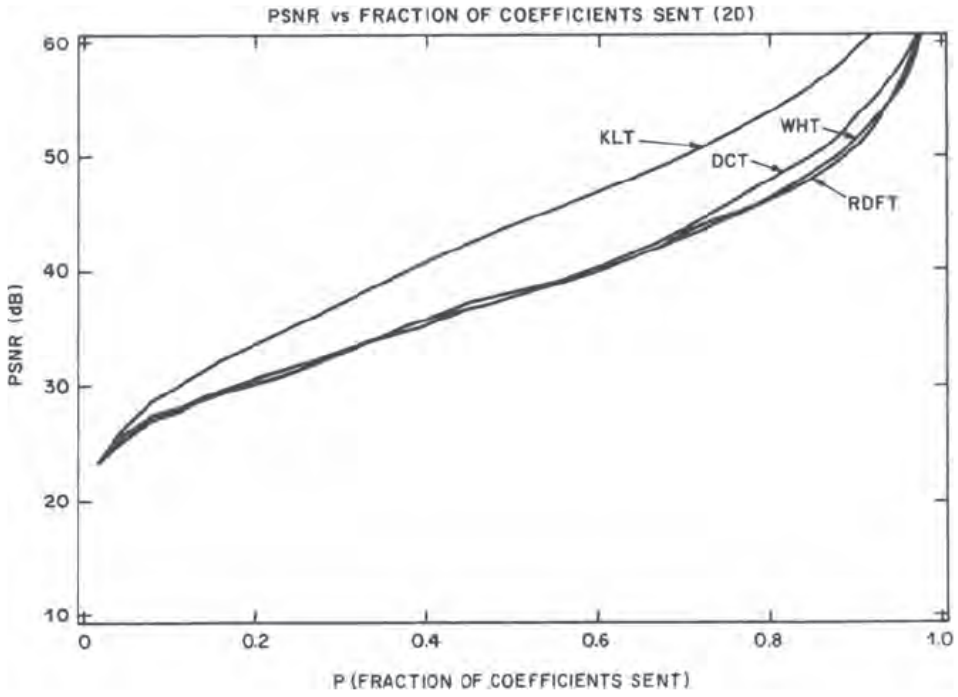


Fig. 5.3.19 Comparison of truncation errors using nonseparable, two-dimensional, 8×8 blocks with the image "Karen". The coefficients having the largest Mean Square Value (MSV) are transmitted.

6.0e+05	4.9e+04	2.3e+04	1.1e+04	4.5e+03	2.6e+03	1.0e+03	2.9e+02
5.9e+03	8.3e+02	4.4e+02	1.9e+02	1.0e+02	6.1e+01	1.9e+01	8.6e+00
5.8e+02	1.5e+02	8.4e+01	5.7e+01	3.1e+01	1.4e+01	7.6e+00	2.9e+00
4.1e+02	7.4e+01	5.0e+01	2.4e+01	1.2e+01	8.4e+00	3.6e+00	1.8e+00
1.7e+01	3.0e+01	2.0e+01	1.1e+01	6.6e+00	4.9e+00	2.4e+00	1.3e+00
1.3e+02	2.6e+01	1.7e+01	1.0e+01	5.6e+00	5.0e+00	2.6e+00	1.3e+00
4.5e+01	8.8e+00	9.0e+00	4.9e+00	3.2e+00	2.9e+00	1.5e+00	9.9e+01
1.4e+02	2.0e+01	1.3e+00	6.9e+00	3.4e+00	3.6e+00	1.5e+00	9.6e+01

Fig. 5.3.20 Coefficient MSV's using the separable 2D-DCT, 8×8 blocks with the image "Stripes", which has large correlation in the vertical direction.

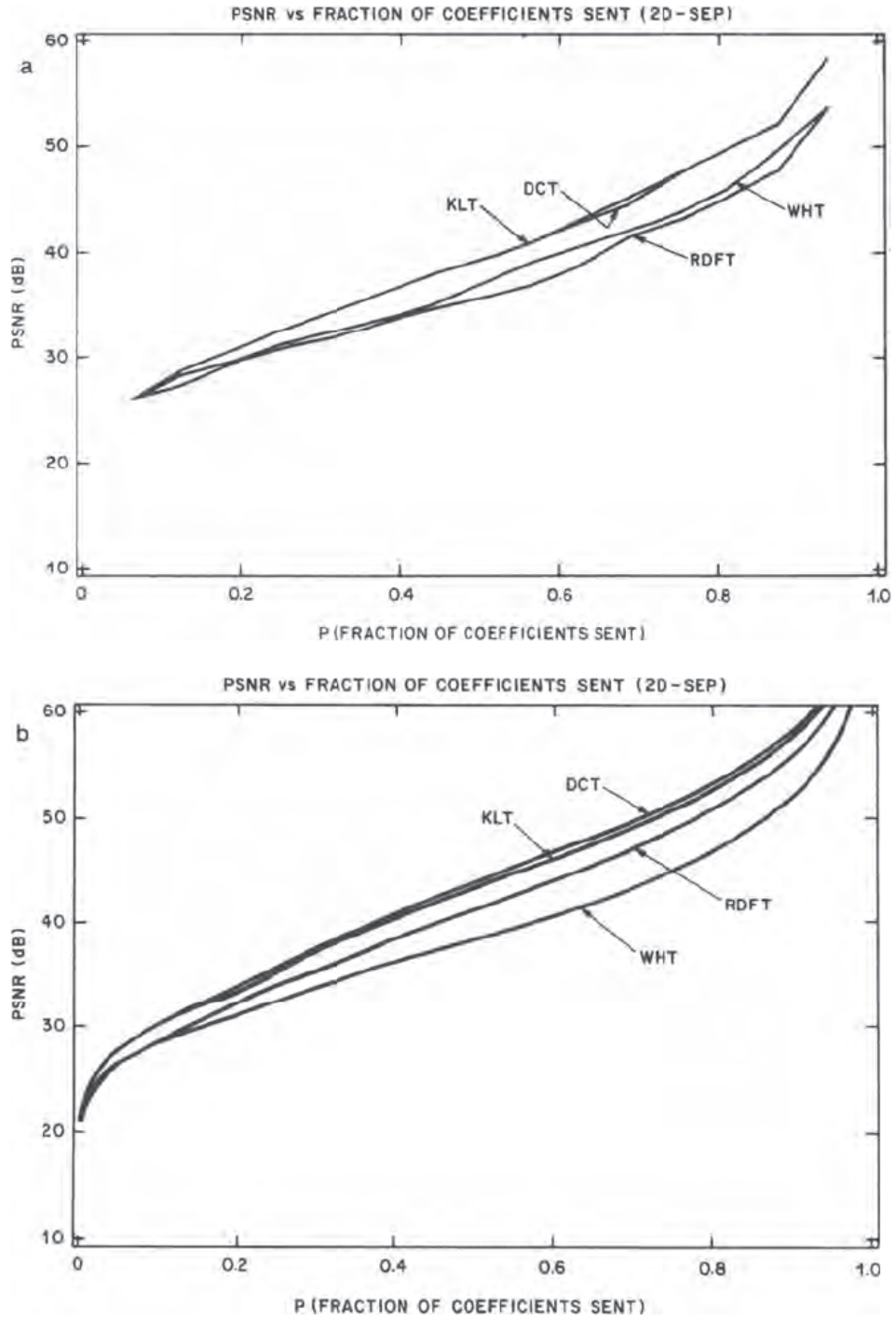


Fig. 5.3.21 Comparison of truncation errors using separable, two-dimensional blocks with the image "Karen". The coefficients having the largest MSV are transmitted. (a) 4×4 blocks, $N = 16$. (b) 16×16 blocks, $N = 256$.

$$\boldsymbol{C} = \boldsymbol{TBT}^t \tag{5.3.56}$$

to obtain an $L \times L$ matrix $\boldsymbol{C}$ containing the transform coefficients. For example, Fig. 5.3.20 shows the coefficient MSV's when the separable DCT (8×8 pel blocks) is applied to the 8-bit PCM data from "Stripes". Note the decreased energy in the vertical spatial frequencies compared with the horizontal frequencies. This reflects the high vertical correlation in this particular picture.

Fig. 5.3.21 shows for $L = 4$ and $L = 16$ the results of separable transformations of $L \times L$ blocks of pels on the picture "Karen" when only pN of the coefficients are transmitted. The KLT was derived from the correlation matrix of one-dimensional $1 \times L$ blocks of pels. We see that the separable DCT performs about as well as the separable KLT. This is probably due to the fact that the KLT cannot adapt very well to nonstationary image statistics in the relatively small block size of $1 \times L$ pels. Also, note that the performance of the separable DCT is only a few dB below that of the nonseparable KLT of Fig. 5.3.19. It is this characteristic behavior that makes the separable DCT the transform of choice in many coding applications.

For various block sizes Fig. 5.3.22 shows PSNR results for $p = 60\%$. As before the energy compaction generally improves as the block size increases. However, the improvement is relatively small for block sizes larger than about 8×8 pels.

The transform block size and shape are important parameters affecting the statistics of the transform coefficients although the picture itself is, of course, the overriding factor. Generally, the larger the block size, the higher the energy compaction achieved by the transform. Also two-dimensional blocks achieve more compaction than one-dimensional blocks. Experience has shown that over a wide range of pictures there is not much improvement in average energy compaction for two dimensional block sizes above 8×8 pels. However, individual pictures with higher nonstationary statistics can always be found for which this rule of thumb is violated (for example, compare the KLT curves of Fig. 5.3.17 and Fig.5.3.19). Also, considerable correlation may remain between blocks, even though the correlation between pels within a block is largely removed.[5.3.21] We shall return to this point in a later section.

5.3.1f Miscellaneous Transforms

Several other transforms have been studied. For example, the Haar Transform[5.3.8] can be computed from an orthogonal (but not orthonormal) matrix $\boldsymbol{T}$ that contains only +1's, −1's and zeros as shown in Fig. 5.3.23. This enables rather simple and speedy calculation, but at the expense of energy compaction performance.

The Slant Transform[5.3.9] has, in addition to the usual constant basis vector $\boldsymbol{t}_1$, a basis vector $\boldsymbol{t}_2$ given (for N even) by

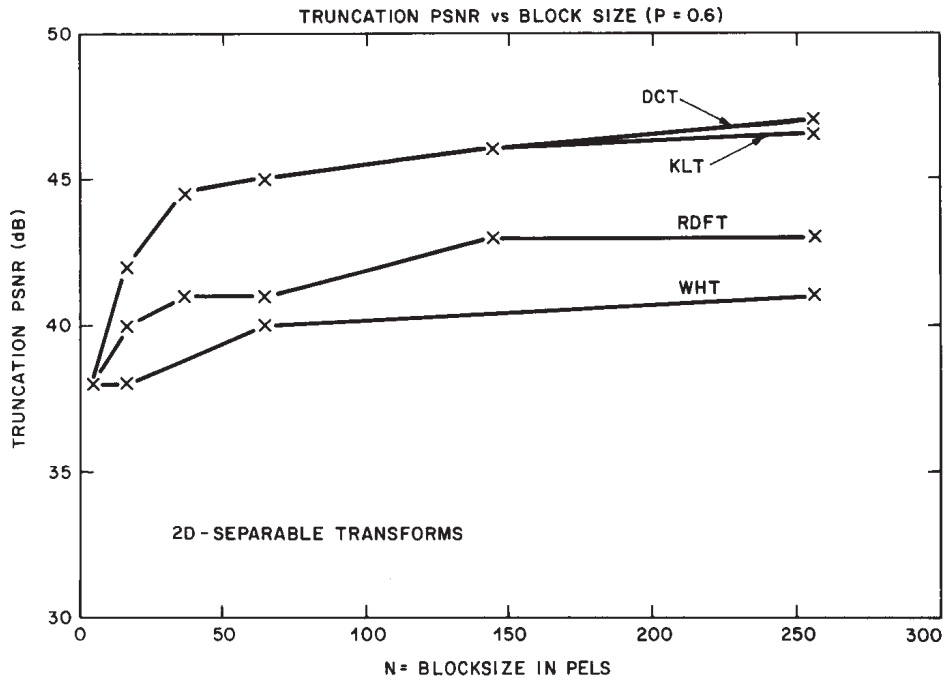


Fig. 5.3.22 Truncation PSNR versus block size for separable transforms with the image "Karen" when 60 percent of the coefficients are kept ($p = 0.6$).

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Fig. 5.3.23 Haar transform matrix. As with the WHT, transformation requires no multiplications.

$$\boldsymbol{t}_2 = \alpha(N-1, N-3, \ldots, 3, 1, -1, -3, \ldots, 1-N)' \qquad (5.3.57)$$

where α is a normalization constant. This vector is thought to better approximate the local behavior of images and thus result in higher energy compaction. However, the $\boldsymbol{t}_2$ of the DCT is very similar, and so is the overall performance in most cases of interest. A fast algorithm has also been developed for the Slant Transform.

The Sine Transform[5.3.10] has matrix elements given by

$$t_{mi} = \sqrt{\frac{2}{N+1}} \sin \frac{mi\pi}{N+1} \qquad (5.3.58)$$

$$m,i = 1 \ldots N$$

Its main utility arises when images are preprocessed and decomposed into a sum of two uncorrelated images, one of which has approximately stationary statistics with a KLT that is approximately the Sine Transform.

The Singular Value Decomposition (SVD)[5.3.11] begins by rewriting the separable inverse transform of Eq. (5.3.11) as

$$\boldsymbol{B} = \boldsymbol{U}'\boldsymbol{C}\boldsymbol{V} \qquad (5.3.59)$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary Lth order transforms. If the basis vectors of $\boldsymbol{U}$ are chosen to be the eigenvectors of $\boldsymbol{B}\boldsymbol{B}'$, and the basis vectors of $\boldsymbol{V}$ the eigenvectors of $\boldsymbol{B}'\boldsymbol{B}$. then the matrix

$$\boldsymbol{C} = \boldsymbol{U}\boldsymbol{B}\boldsymbol{V}' \qquad (5.3.60)$$

will be diagonal with $\{c_{mm}^2 = \lambda_m,\ 1 \le m \le L\}$ where $\lambda's$ are the L distinct eigenvalues resulting from the above operations. Eq. (5.3.12) then becomes

$$\boldsymbol{B} = \sum_{m=1}^{L} \lambda_m \boldsymbol{u}_m \boldsymbol{v}_m' \qquad (5.3.61)$$

If the λ's are arranged in decreasing order and only the first pN of them are significant ($p < 1$), then specification of $\boldsymbol{B}$ requires the transmission of

$$\{\lambda_m,\ \boldsymbol{u}_m,\ \boldsymbol{v}_m,\ 1 \le m \le pN\} \qquad (5.3.62)$$

So far, SVD has found more use in image restoration than in image coding.

5.3.2 Quantization of Transform Coefficients

We have seen that transform coding does a reasonably good job of exploiting *statistical* redundancy in images by decorrelating the pel data and by compacting information into the lower order coefficients. We now present a framework for the exploitation of *subjective* redundancy in pictures using transform coding.

Unlike DPCM, where a particular quantized sample affects relatively few pels in the reconstructed picture, quantization (or deletion) of a particular transform coefficient affects every pel in the transformed block. This *spreading* of the quantization error often makes the design of transform coefficient quantizers quite different than that of DPCM quantizers.

For example, consider the case where we employ a fairly sizable two-dimensional block as shown in Fig. 5.3.24. In many cases, part of the block will be low-detail flat area, and part will be high-detail busy area. Excessive quantization error in one coefficient will eventually cause a distortion pattern to become visible in the flat area, whose spatial frequency corresponds to the basis vector of the coefficient. Because of spatial masking, the distortion will probably not be visible in the busy area of the block.

If quantization error is to be kept below the visibility threshold and if the spatial frequencies of the basis vectors are far enough apart from each other that they are detected independently by the human visual system, then an appropriate quantization strategy might be

$$|c_m - \tilde{c}_m| < T_m \tag{5.3.63}$$

where T_m is the visibility threshold (see Fig. 4.3.11a) corresponding to basis vector $\boldsymbol{t}_m$.

However, for large block sizes the basis vector frequencies are quite close to each other, and they are not detected independently. In this case we might invoke the bandpass filter model of human vision (see Fig. 4.3.10) and partition the basis vectors into sets $\{S_i\}$ of similar spatial frequencies. For subthreshold distortion, the quantization strategy might be designed to satisfy*

$$\sum_{m \in S_i} (c_m - \tilde{c}_m)^2 < T_i \qquad \forall i \tag{5.3.64}$$

where T_i is the detection threshold corresponding to the spatial frequencies in S_i.

---

* Eq. (5.3.64) is only an approximate definition of nondetection by a human observer. Basis vectors having similar spatial frequencies combine in a complicated way, and an exact relationship for detectability is not yet known.

Eq. (5.3.64) applies to coefficients that have sizable MSV's, and in this case $E(c_m - \tilde{c}_m)^2$ is the mean square quantization error. However, we saw earlier that the MSV's of some coefficients may be so small that they need not be transmitted at all. In this case, the mean square error is $\sigma_m^2 = E(c_m^2)$, and is referred to as the *truncation error* due to that coefficient not being transmitted.

If a certain percentage of the low energy transform coefficients are set to zero and not transmitted in order to conserve bit-rate, then as the number of zeroed coefficients increases, distortion is first detected, not in the low-detail flat area, but in the high-detail busy area of the block or at the boundary between the two areas. The general effect is a blurring or loss of sharpness in regions of rapid luminance variation. For this reason, in Eq. (5.3.64), the threshold values corresponding to high spatial frequencies (low MSV) may have to be reduced somewhat below the values obtained from Fig. 4.3.11a, in order to preserve the desired degree of resolution in high-detail areas of the image.

Transform coefficient c_1 corresponds to a spatial frequency of zero, for which Fig. 4.3.11a does not apply. Quantization error in c_1 first appears at the block boundary and can cause a very visible discontinuity between adjacent blocks, especially in low detail areas of the image. The visibility of these discontinuities will determine the appropriate threshold value of T_1. Depending on the block size, quantization error in other low frequency coefficients may also contribute to these *block edge effects*, and this may require further adjustment of thresholds. The DCT and KLT generally have less block edge effects than other unitary transforms due to the fact that they have more slowly varying basis vectors.

In some cases, especially with small block sizes, a given block may contain little or no low-detail flat areas. Then, spatial masking by the high-detail luminance variations will allow for a larger quantization error than indicated in Eq. (5.3.64). We can take this into account for each block by defining a *Spatial Activity Function* A_F, as discussed in Section 4.3.1b, which measures the relative amount of detail in the block. A_F could be defined using gradient magnitudes as suggested in Chapter 4. However, in transform coding a more convenient definition often utilizes the non-DC transform coefficients. For example,

$$A_F = 1 + q \sum_{m=2}^{N} c_m^2 \tag{5.3.65}$$

where q is a suitable normalization constant. Alternatively, a subset of the non-DC coefficients might be used to simplify calculations somewhat, or instead of a direct summation of c_m^2, a weighted sum might be employed. In any event, using a spatial activity function the constraint on quantization error for subthreshold distortion becomes

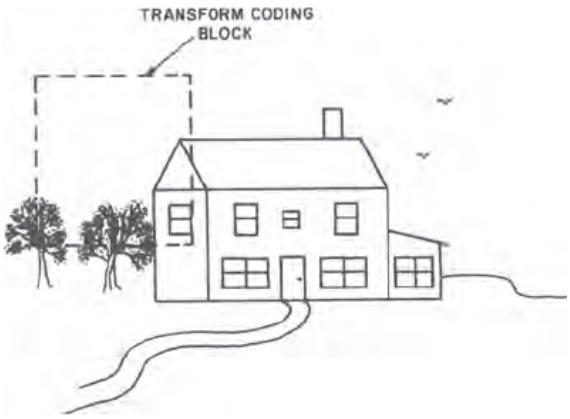


Fig. 5.3.24 In many cases, part of a transform coding block will be low-detail flat area, and part will be high-detail busy area.

Fig. 5.3.25 Coded image "Karen" using the separable 2D-DCT with 8×8 blocks, uniform quantization and entropy coding. The coefficients having the largest MSV are sent. PSNR's and entropies are (left-right, top-bottom):
(a)PSNR = 38.0dB, 1.24 bits/pel (b)PSNR = 33.5dB, 0.67 bits/pel
(c)PSNR = 29.9dB, 0.36 bits/pel (d)PSNR = 25.7dB, 0.13 bits/pel

$$\sum_{m \in S_i} \frac{(c_m - \tilde{c}_m)^2}{A_F} < T_i \quad \forall i \tag{5.3.66}$$

or dividing through by T_i and defining $T_m = T_i$ for $m \in S_i$,

$$\sum_{m \in S_i} \frac{(c_m - \tilde{c}_m)^2}{A_F T_m} < 1 \qquad \forall i \tag{5.3.67}$$

If the bit-rate provided for transmission is not high enough to allow for sub-threshold distortion, then quantization must be made coarser and distortion will be above the threshold of visibility. In this case, we would like to spread the distortion over all of the spatial frequency bands in an effort to randomize it so that, in flat areas at least, it appears to a human observer to be like random noise. Thus, we relax Eq. (5.3.67) to get

$$\sum_{m \in S_i} \frac{(c_m - \tilde{c}_m)^2}{A_F T_m} < \left[1 + \varepsilon\right] \qquad \forall i \tag{5.3.68}$$

where ε is either zero or positive depending on whether the distortion is sub- or supra-threshold, respectively.

The quantization/coding task can then be stated as either:

1) For a fixed ε minimize the average bit-rate, or

2) for a fixed average bit-rate minimize ε.

The possible solutions to this coding problem have only been partially studied.[5.3.12] Here we simply state that they depend intimately on block-size, the particular transform used, quantization algorithm, buffer strategy, etc.

The quantization problem can be (and in practice usually is) simplified considerably by defining a block distortion

$$d = \frac{1}{N} \sum_{m=1}^{N} \frac{(c_m - \tilde{c}_m)^2}{A_F T_m} \tag{5.3.69}$$

and an average block distortion*

* As with DPCM, in some cases the statistical averaging operator E may be profitably replaced by a subjectively weighted average.

$$D = \frac{1}{N} \sum_{m=1}^{N} E\left[\frac{(c_m - \tilde{c}_m)^2}{A_F T_m}\right] \tag{5.3.70}$$

This is a noise visibility and spatial frequency weighted MSE distortion criterion, where the spatial activity function A_F is a measure of the masking and $1/T_m$ is a measure of the visibility of various spatial frequencies, e.g., see Fig. 4.3.11b in Chapter 4.

Consider, for now, only blocks having similar values for the spatial activity function A_F. These blocks would all be coded in the same way by adaptive systems. The distortion for this class of blocks is given by

$$D \approx \frac{1}{N} \sum_{m=1}^{N} \frac{\varepsilon_m}{A_F T_m} \tag{5.3.71}$$

where $\varepsilon_m \triangleq E[(c_m - \tilde{c}_m)^2]$ is the mean square quantization error (MSE) for coefficient c_m. Let r_m be the average number of bits used to code c_m. Then the average block bit-rate is

$$R = \frac{1}{N} \sum_{m=1}^{N} r_m \quad \text{bits/pel} \tag{5.3.72}$$

We saw in Chapter 3 that the functional relationship $\varepsilon_m(r_m)$ between coefficient MSE and bit-rate depends on the probability distribution (PD) of the coefficient c_m and on the quantization/coding strategy employed, e.g., uniform with fixed word-lengths, Lloyd-Max with fixed word-lengths, uniform with entropy coding, etc. Assuming $\varepsilon_m(\cdot)$ is a continuous function of its argument, we now show that for R fixed, a necessary condition to minimize D is (here prime denotes first derivative, not transpose):

$$\frac{\varepsilon_m'(r_m)}{A_F T_m} = \begin{cases} -\theta & r_m > 0 \\ s_m \geq -\theta & r_m = 0 \end{cases} \tag{5.3.73}$$

where θ is a nonnegative parameter. The proof is by contradiction. Suppose $r_m, r_n > 0$ and

$$\frac{\varepsilon_m'(r_m)}{A_F T_m} > \frac{\varepsilon_n'(r_n)}{A_F T_n} \tag{5.3.74}$$

Then reducing r_m by Δ and increasing r_n by Δ would leave R unchanged. However, D would undergo a net decrease, which contradicts our original assumption. Similarly, for $r_m > 0, r_n = 0$.

Eq. (5.3.73) is a fundamental result of transform coding theory for a frequency weighted MSE distortion measure. It tells us how to apportion the bits amongst the transform coefficients. If the MSE functions $\varepsilon_m(\cdot)$ are discontinuous, e.g., defined only by integer bit-rates, the problem is much more complicated,[5.3.13, 5.3.14] although approximate solutions can often be obtained by solving the appropriate continuous problem.

In Chapter 3 we saw that all transform coefficients except c_1 had approximately a Laplacian Probability Distribution (PD). Coefficient c_1 had a PD that could only be approximated by a uniform distribution, unless the block size was very large, in which case Gaussian was better. Also, quantization error in c_1 is not very well represented subjectively by MSE. For these reasons, c_1 is often quantized uniformly and coded using fixed word lengths.

Two quantization schemes are commonly used for the non-DC transform coefficients. They are 1) Lloyd-Max quantization with fixed-length code words and 2) Uniform quantization with variable-length code words, i.e., near optimum entropy coding. With Lloyd-Max quantization of zero mean coefficients having Laplacian PD's we saw in Chapter 3 that the MSE ε_m (which is in fact discontinuous) could be approximated using the piecewise linear relation

$$\ln \frac{\varepsilon_m(r_m)}{\sigma_m^2} \approx \begin{cases} -0.84 r_m & r_m < 2 \text{bits} \\ 0.90 - 1.29 r_m & r_m \geq 2 \text{bits} \end{cases} \tag{5.3.75}$$

where σ_m^2 is the MSV of c_m. This can be easily differentiated and substituted into Eq. (5.3.73) to give

$$\frac{\varepsilon_m}{A_F T_m} = \begin{cases} \dfrac{\theta}{1.29} & r_m \geq 2 \\ \dfrac{\theta}{0.84} & 0 < r_m < 2 \\ \dfrac{\sigma_m^2}{A_F T_m} & r_m = 0 \end{cases} \tag{5.3.76}$$

For a given value of the parameter θ we find the coefficient bit assignment by using Eq. (5.3.75) to solve the first of the above equations for r_m and seeing if it is 2 bits or more. If not, we solve the second and see if r_m is positive. If not, we set $r_m = 0$ and do not transmit that particular coefficient. This gives us the coefficient bit assignments for Lloyd-Max quantization with fixed word-length coding.

If the quantized transform coefficients are to be coded independently of each other then we must round each r_m to the nearest integer. However, in certain cases combined coding may be profitable. For example, if two coefficients are quantized with three and five levels, respectively, then they can be coded together (15 possible combinations) using only a 4-bit word.

With uniform quantization and entropy coding of Laplacian coefficients, we saw in Chapter 3 that for a given MSE ε_m could be approximated by

$$\ln \frac{\varepsilon_m(r_m)}{\sigma_m^2} \approx 0.19 - 1.39 r_m \tag{5.3.77}$$

where the entropy r_m is in bits/sample and the quantizer contains the zero† level. Differentiating and substituting into Eq. (5.3.73) we get

$$\frac{\varepsilon_m}{A_F T_m} = \begin{cases} \dfrac{\theta}{1.39} & r_m > 0 \\ \dfrac{\sigma_m^2}{A_F T_m} \le \dfrac{\theta}{1.39} & r_m = 0 \end{cases} \tag{5.3.78}$$

This important result says that every coefficient that is transmitted should have the same weighted mean square error $\theta/1.39$, whereas untransmitted coefficients should each have a smaller weighted mean square error.

As before, for a given θ we use Eq. (5.3.77) to solve the first of the above equations for r_m, i.e.,

$$r_m = \frac{1}{1.39}\left[0.19 + \ln \frac{1.39\sigma_m^2}{\theta A_F T_m}\right] \tag{5.3.79}$$

$$\approx \frac{1}{1.39} \ln \frac{1.68\sigma_m^2}{\theta A_F T_m} \text{ bits}$$

and see if it is positive.* If not, we set $r_m = 0$ and do not transmit the

† For a given interval size, mid-step quantization gives a smaller entropy than mid-riser quantization.

* In fact, $r_m \ge 1$ may be a more practical constraint since $r_m < 1$ requires block coding and may not be worth the added complexity.

coefficient. This gives us the coefficient bit assignments $\{r_m\}$ for uniform quantization with entropy coding.#

Suppose now that the coefficients are ordered according to decreasing σ_m^2/T_m and, utilizing Eq. (5.3.78), a certain fraction $p < 1$ of them are transmitted. (Note that a very good approximation to this ordering is to arrange the coefficients according to increasing spatial frequency of their basis vectors.) The relationship between θ and p can be estimated from Eq. (5.3.79) and the constraint

$$r_{pN} \geq 0 \geq r_{pN+1} \tag{5.3.80}$$

If the block size is not too small, we may assume $r_{pN} \gtrsim 0$ and from Eq. (5.3.79)

$$\theta \lesssim \frac{1.68\sigma_{pN}^2}{A_F T_{pN}}$$

and (5.3.81)

$$r_m \approx \frac{1}{1.39} \ln \frac{\sigma_m^2 T_{pN}}{\sigma_{pN}^2 T_m} \qquad m < pN$$

Thus,

$$R \approx \frac{1}{1.39N} \sum_{m=1}^{pN} \ln \frac{\sigma_m^2\ T_{pN}}{\sigma_{pN}^2\ T_m} \text{ bits/pel} \tag{5.3.82}$$

From Eq. (5.3.78) the weighted MSE distortion D then becomes

$$D \triangleq \frac{1}{N} \sum_{m=1}^{N} \frac{\varepsilon_m}{A_F T_m} \tag{5.3.83}$$

$$= \frac{p\theta}{1.39} + \frac{1}{N} \sum_{m>pN}^{N} \frac{\sigma_m^2}{A_F T_m}$$

Often, coefficient c_1 is positive and uniformly distributed, in which case Eq. (5.3.79) gives an r_1 that is too large by about 1.1 bits/pel. However, the discrepancy is small; and moreover it is on the conservative side.

That is, the distortion is the sum of two terms, the first being the *Quantization* error and the second being the *Truncation* error.

These two types of errors have a markedly different appearance in coded images having supra-threshold distortion. Quantization error is noiselike, whereas truncation error causes a loss of resolution. The tradeoff between these two types of distortion is not well understood, and depends to a large extent on the application, e.g., surveillance, television, etc. Thus, in practice, these equations must be used with care and the thresholds $\{T_m\}$ must be determined experimentally.

Note the similarity between Eqs. (5.3.78) and (5.3.68). In both cases, distortion is spread uniformly in all spatial frequency bands that are transmitted. Thus, even though the weighted MSE distortion measure of Eq. (5.3.69) does not take into account the way the error is distributed among the spatial frequencies, the subsequent optimization procedure nevertheless ends up distributing the error evenly over the transmitted frequencies.

An important corollary of Eq. (5.3.78) stems from the relationship derived in Chapter 3 between the MSE ε_m and the uniform quantizer step size Δ_m. We saw from Eq. (3.2.38) that for reasonable bit-rates, e.g., $r_m \geq 1$,

$$\Delta_m \approx \sqrt{12\varepsilon_m} \tag{5.3.84}$$

which is independent of the MSV of c_m. Thus, from Eq. (5.3.78), assuming $0 < r_m < 1$ is not allowed, we may write the uniform quantizer step-size directly as

$$\Delta_m = \sqrt{\frac{12A_F T_m \theta}{1.39}} \quad m < pN \tag{5.3.85}$$

Note that if T_m is assumed constant, then the same quantizer may be used for all transform coefficients. Alternatively, the coefficients c_m could be normalized by $\sqrt{T_m}$ prior to quantization.

However, the probability distribution of the quantizer outputs will vary considerably for coefficients having different MSV's. Thus, for optimum performance different variable-word-length codes should be used for coefficients having dissimilar variances.

At this point it is worthwhile to recapitulate these results for uniform quantization with entropy coding. If the coefficients are ordered according to decreasing σ_m^2/T_m, and a fraction $p < 1$ of them are sent, then the optimum quantization step size for c_m is given by Eq. (5.3.85), the distortion by Eq. (5.3.83), and the average bit-rate by Eq. (5.3.82).

Fig. 5.3.25 shows the picture "Karen" transform coded at several bit rates using the DCT with 8×8 blocks, uniform quantization and entropy coding. For simplicity, A_F and T_m were set to unity for all blocks, i.e., simple MSE was

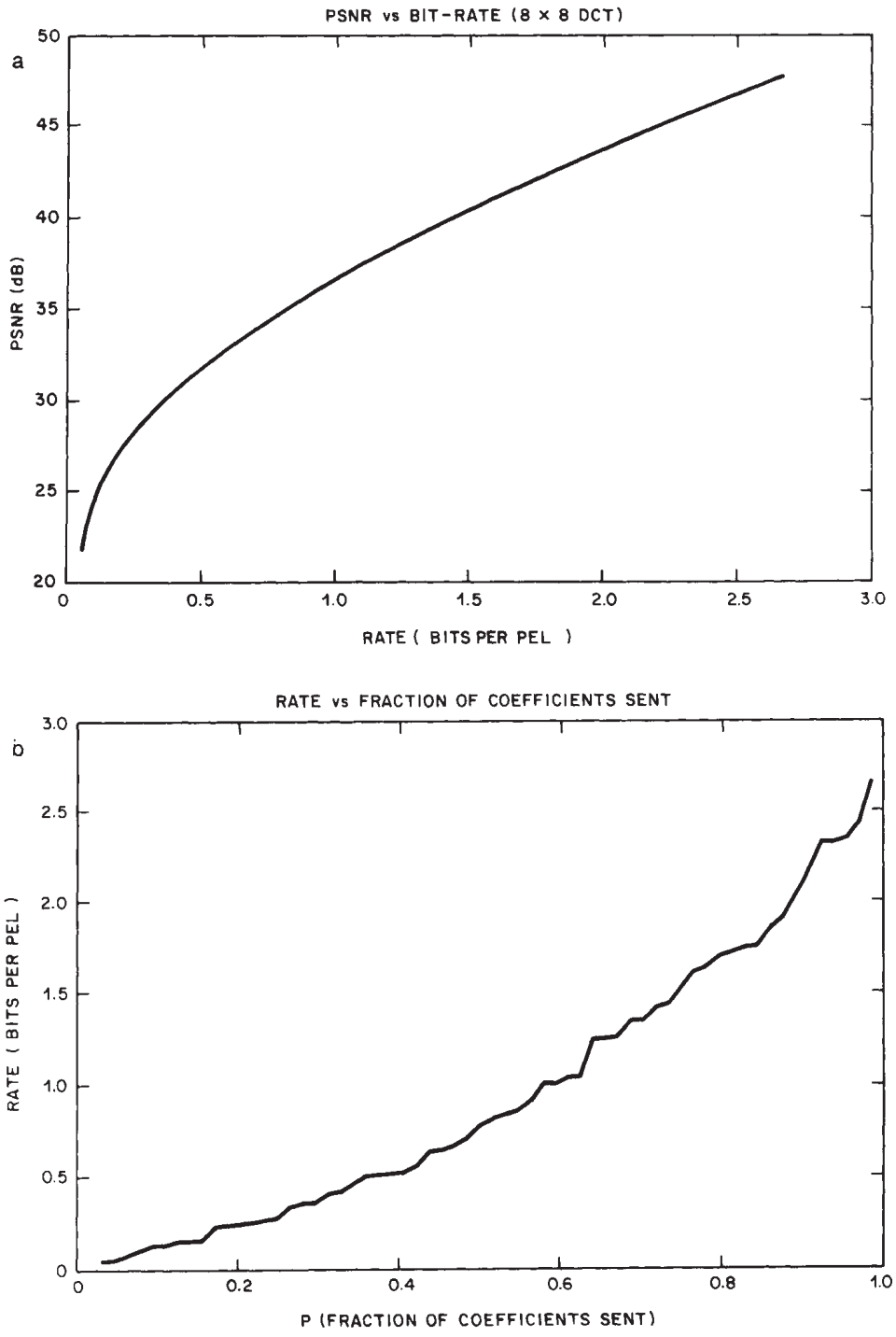


Fig. 5.3.26 Relationship between PSNR, bit-rate (R) and fraction (p) of transmitted coefficients of the images coded in Fig. 5.3.25 (a) PSNR vs. R, (b) R vs. p.

used as a distortion criterion. Also, $r_m < 1$ was not allowed.

Fig. 5.3.26 shows PSNR versus entropy R for the same picture. Also shown for each value of R is the fraction p of coefficients that are transmitted. Note the nearly linear relationship between R and p that exists over most of the range.

Similar results can be derived using the mathematical models defined in Chapter 3. For (real) separable 2D-transformation of $L \times L$ pel blocks $\boldsymbol{B} = [b_{ij}]$, the coefficient MSV's become [see Eq. (5.3.50)]

$$\sigma_{mn}^2 = \sum_{i\,j\,k\ell} r_{i\,j\,k\ell}\, t_{mi}\, t_{nj}\, t_{mk}\, t_{n\ell} \tag{5.3.86}$$

where the correlation $r_{i\,j\,k\ell} \triangleq E(b_{ij}, b_{k\ell})$. From these MSV's (arranged in decreasing order or increasing spatial frequency), the relation between rate R and distortion D can be found from Eqs. (5.3.78-83). For example, using an isotropic model with 8-bit PCM data

$$r_{i\,j\,k\ell} = \frac{255^2}{3}\, \rho^{\sqrt{(i-k)^2+(j-\ell)^2}} \tag{5.3.87}$$

Fig. 5.3.27 shows 2D-DCT PSNR versus R for $\rho = 0.98$ and various values of L (again for $A_F T_m = 1$).

5.3.3 Adaptive Coding of Transform Coefficients

If the pictures to be coded have fairly similar statistics and the transform block-size is reasonably large, then the transform coefficient statistics will not change much from block to block. For example, aerial photos of urban regions might fall into this category. In this case, coefficient MSV's can be measured, picture quality determined, quantization parameters calculated according to the zonal sampling techniques of the previous Section, and the bit-rate can be estimated with high accuracy. This approach we term as *Nonadaptive* transform coding.

However, if image statistics vary widely and/or the block-size is relatively small so that some blocks are in low-detail areas of the picture while others are in high-detail areas, then nonadaptive techniques will often be unsatisfactory. For example, if the coefficients of each block are quantized using Lloyd-max quantization, and each coefficient is assigned a fixed number (possibly 0) of bits using Eq. (5.3.75) with $A_F = 1$ and some set of assumed coefficient MSV's, then the number of bits per block is constant, and very little data buffering is required. However, this simple zonal sampling approach is usually not much more efficient than DPCM in terms of bit-rate and picture quality. Moreover, it is far more complex and expensive to implement than DPCM.

The situation improves markedly if *Adaptive* transform coding techniques are employed. For example, using a technique called threshold sampling, only

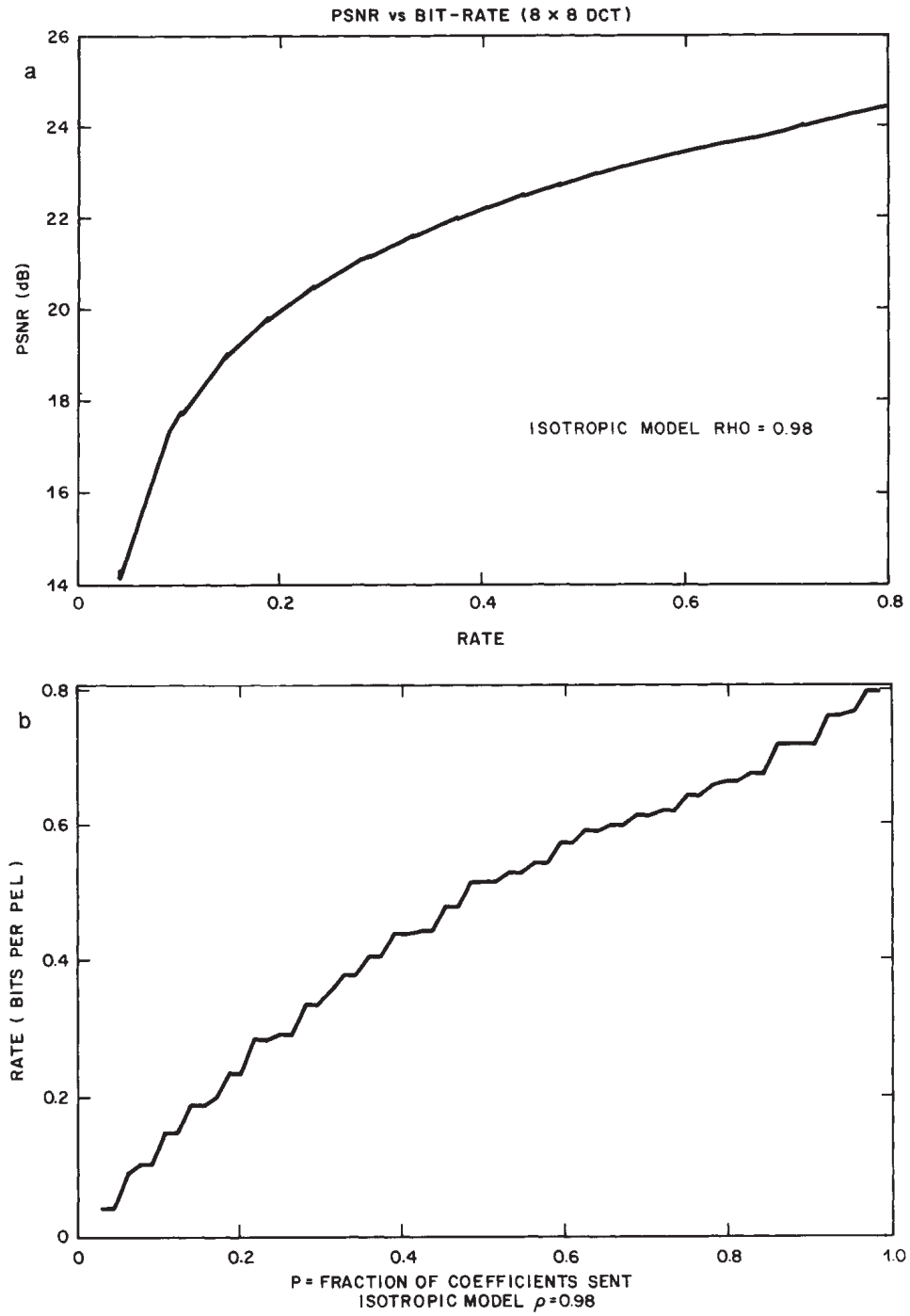


Fig. 5.3.27 Relationship between PSNR, R and p for the isotropic correlation model with adjacent pel correlation $\rho = 0.98$ and separable 2D-DCT coding of 8×8 blocks. (a) PSNR vs. R, (b) R vs. p.

those coefficients in each block whose magnitude exceeds a prescribed threshold are transmitted. With these methods low-detail blocks, having a large number of insignificant coefficients, are coded with far fewer bits than the higher-detail blocks, resulting in a significantly smaller bit-rate. Also, in order to exploit masking effects, high-detail blocks are coded with more quantization error than low detail blocks where masking does not occur and block boundary distortion is more visible. Indeed the performance of transform coding in general becomes attractive compared with DPCM only if adaptive techniques are used and the proportion of low-detail blocks is sizable, so that the overall bit-rate can be reduced.

A drawback of adaptive transform coding is an increased sensitivity to transmission bit errors. The effect of these can propagate over sizable areas of the received picture not only because each erroneous coefficient affects all pels of the block, but also because sending a variable number of coefficients and possibly using variable word-length codes makes it possible for the decoder to lose synchronization. Once this happens the receiver may decode too many or too few coefficients in several successive blocks. Also, blocks may be deleted or extra ones may be inserted into the received picture.

Adaptive coding, of sorts, occurs if we use uniform quantization plus entropy coding as defined in Eq. (5.3.78). Recall that here, we chose quantizer parameters according to an assumed A_F, some set of assumed coefficient MSV's and a value of θ that provides acceptable image quality. However, low-detail blocks will generally have coefficient MSV's that are much smaller than those assumed, and for optimum coding according to Eq. (5.3.79) the coefficients should be coded at lower bit-rates. Also, many more of them should be assigned $r_m = 0$ and not transmitted at all. However, entropy coding automatically reduces the bit-rate in this case since smaller coefficient values are assigned shorter code words, and if the MSV's decrease, so will the average code-word lengths. Moreover, if mid-step quantizers are used, then the smallest coefficients will be quantized to zero, and if these values are run-length coded their bit-rates are markedly reduced, although not to zero. Thus, with entropy coding, the bit-rate can indeed decline significantly for low-detail blocks having relatively small coefficient variances.

However, this scenario is not optimum. For one thing, a Huffman code designed for one probability distribution is not necessarily optimum for another. Also, in low-detail blocks the percentage p of transmitted (or nonzero) coefficients declines, and the quantizer step size Δ_m of Eq. (5.3.85) should be changed. However, offsetting this to some extent is the fact that low-detail blocks should also use a smaller value for the activity function A_F.

A more nearly optimum procedure is to *classify* each transform block according to its spatial activity into one of say L classes $\{C_\ell, \ell = 1...L\}$. Within each class C_ℓ the coefficient MSV's $\{{}_\ell\sigma_m^2, m = 1...N\}$ and the activity function ${}_\ell A_F$ should be about the same for each block. Thus, a nearly optimum

code can be designed for each class such that the distortion is equalized over all classes. For example, with uniform quantization and entropy coding, Eqs. (5.3.78-85) may be used for each class to derive the percentage p_ℓ of transmitted coefficients and the step size $\{{}_\ell\Delta_m\}$ of the uniform quantizer, so that the average distortion

$$D = \frac{p_\ell \theta}{1.39} + \frac{1}{N} \sum_{m>Np_\ell}^{N} \frac{{}_\ell\sigma_m^2}{{}_\ell A_F T_m} \tag{5.3.88}$$

is the same for all classes. For class C_ℓ the average bit-rate, from Eq. (5.3.79), then becomes

$$R_\ell = \frac{1}{1.39N} \sum_{m=1}^{Np_\ell} \ln \frac{1.68\,{}_\ell\sigma_m^2}{{}_\ell A_F T_m \theta} \tag{5.3.89}$$

If a particular image has a fraction w_ℓ of its transform blocks in class C_ℓ, then the overall average bit-rate is

$$R = \sum_{\ell=1}^{L} w_\ell R_\ell \tag{5.3.90}$$

Using the classification method of adaptive transform coding, the receiver either must be told or must be able to derive which class each block belongs to. If the block size is not too small and the number of classes not too large, then this *overhead* information can be transmitted directly without much penalty. Otherwise, alternative methods must be devised. For example, with small block sizes, spatial activity might be determined from the non-DC coefficients of adjacent, previously transmitted blocks.

So far in this Section, we have assumed that the allowable distortion D is fixed and that the average bit-rate R can vary in whatever way is necessary to achieve that distortion. This arrangement may be suitable for computer disk storage, facsimile transmission, store and forward services, etc. However, in many other applications, e.g., television, not only is the channel data-rate constant, but also the number of images (frames) per second that must be displayed is fixed. This means that if the advantages of adaptive coding are to be exploited, then data buffers must be provided at both the transmitter and receiver in order to smooth the data rate prior to transmission. The size, i.e., storage capacity, of these buffers is a major factor in determining the degree of adaptability and coding efficiency of the system. If the buffer size is too small, then the opportunity for bit-rate trade-off between low-detail and high-detail regions of the picture is severely restricted, and the required channel rate is determined essentially by the number of bits per block in high-detail regions. If the buffer is too large the transmission delay may be excessive for interactive

communication applications such as videotelephone.

With large buffers and "normal" pictures having the assumed statistics, considerable improvement results from smoothing of the bit-rate between low- and high-detail regions of the picture. However, provision must also be made in any adaptive system for the occasional arrival of "abnormal" pictures that have large areas of high-detail and, with fixed distortion coding as described above, would produce too high a bit-rate to be transmitted without buffer overflow.

Abnormal pictures present two problems. First their presence must be detected, and second something must be done to reduce the bit-rate produced by the coder. Detection of too high a data rate is often accomplished by keeping track of how full the transmitter buffer is. When buffer overflow threatens, the coder bit-rate must be reduced, i.e., the distortion increased.

With transform coders the most effective way of decreasing the bit-rate is to reduce the number of transmitted coefficients. Indeed with the separable DCT, uniform quantization and entropy coding, we saw in Fig. 5.3.26 that the bit-rate R was approximately proportional to the fraction p of coefficients transmitted. This behavior is a direct consequence of the way the mean square coefficient values decline with spatial frequency. For example, suppose the percentage p of transmitted coefficients is increased by

$$\Delta p \triangleq \frac{1}{N} \tag{5.3.91}$$

and that θ is changed by $\Delta\theta$ in order to maintain optimal coding. Then from Eq. (5.3.79)

$$\Delta R \approx \frac{\Delta p}{1.39N} \ln \frac{1.68\ \sigma_{pN+1}^2}{\theta A_F\ T_{pN+1}} - \frac{p}{1.39} \frac{\Delta\theta}{\theta} \tag{5.3.92}$$

If we assume from Eq. (5.3.80) that the first term is small, then

$$\frac{\Delta R}{\Delta p} \approx - \frac{p}{1.39\theta} \frac{\Delta\theta}{\Delta p} \tag{5.3.94}$$

We saw in Fig. 5.3.15 and Fig. 5.3.16 that for the 2D-DCT, the coefficient MSV's exhibited an approximate linear behavior on a log-log plot over a wide range of spatial frequencies.

If this same behavior were to extend to the frequency weighted MSV's, i.e.,

$$\ln \frac{\sigma_{pN}^2}{A_F\ T_{pN}} \approx a_1 - a_2 \ln p \tag{5.3.95}$$

then from Eq. (5.3.81) we have

$$\ln \theta \approx \ln 1.68 + a_1 - a_2 \ln p$$

$$\triangleq \ln a_3 - a_2 \ln p \tag{5.3.96}$$

Differentiating we obtain

$$\frac{\Delta\theta}{\Delta p} \approx - a_2 \frac{\theta}{p} = -a_2 a_3 p^{-(1+a_2)} \tag{5.3.97}$$

Substituting in Eq. (5.3.94)

$$\frac{\Delta R}{\Delta p} \approx \frac{a_2}{1.39} \tag{5.3.98}$$

which is the observed linear relationship between R and p.

If we let $\Delta p = -1/N$ then the change in distortion is from Eq. (5.3.83)

$$\Delta D = \frac{\Delta p \theta + p \Delta\theta}{1.39} + \frac{1}{N} \frac{\sigma_{pN}^2}{A_F T_{pN}} \tag{5.3.99}$$

which from Eq. (5.3.81)

$$\approx \frac{\Delta p \theta + p \Delta\theta}{1.39} - \Delta p \frac{\theta}{1.68}$$

which from Eq. (5.3.97)

$$\approx \left[0.124 - \frac{a_2}{1.39}\right] \theta \, \Delta p$$

Thus, from Eq. (5.3.96)

$$\frac{\Delta D}{\Delta p} \approx - a_3 \left[\frac{a_2}{1.39} - 0.124\right] p^{-a_2} \tag{5.3.100}$$

The above equations provide a theoretical basis for a buffer control strategy when 2D-DCT coding is used with uniform quantization and entropy coding. If the transmitter buffer threatens to become too full or too empty, the percentage of transmitted coefficients is changed by Δp, and the quantizer step size is changed by [differentiate Eq. (5.3.85)]

$$\Delta(\Delta_m) \approx \frac{\Delta\theta}{2\Delta_m} \cdot \frac{12A_F T_m}{1.39} \tag{5.3.101}$$

This results in a change in data rate of ΔR and a change in distortion of ΔD.

If adaptive transform coding is carried out by classification of blocks according to spatial activity, then buffer control is somewhat more complicated. In this case, each class C_ℓ has parameters p_ℓ and R_ℓ that must be optimized, and the simplest approach is usually to change θ in order to control rate and distortion. If buffer overflow or underflow threatens, then the p_ℓ are changed in such a way that the new distortion $D+\Delta D$ is the same in all classes. For example, with uniform quantization and entropy coding, for a given $\Delta\theta$ the new quantizer step sizes are found from Eq. (5.3.101). Then

$$\Delta\theta \approx -\, a_2 \frac{\theta_\ell}{p_\ell} \Delta p_\ell \tag{5.3.102}$$

determines Δp_ℓ for each class. Then*

$$\Delta D \approx -\, a_3 \left[\frac{a_2}{1.39} - 0.124\right] p_\ell^{-a_2} \Delta p_\ell \tag{5.3.103}$$

The change in bit-rate for each class is [from Eq. (5.3.98)]

$$\Delta R_\ell \approx \frac{a_2}{1.39} \Delta p_\ell \tag{5.3.104}$$

and if a particular image has a fraction w_ℓ of its blocks in class C_ℓ, then the overall change in bit-rate is

$$\Delta R = \sum_{\ell=1}^{L} w_\ell \, \Delta R_\ell \tag{5.3.105}$$

The above techniques perform optimum coding under the assumed distortion criterion. However, there are countless suboptimal algorithms that have also been studied and whose performance may be adequate in some circumstances. For example, the quantization might be nonadaptive from block-to-block, whereas the percentage p of coefficients transmitted could vary according to spatial activity. Also, we would like to send only the coefficients

* Note that a_2 and a_3 may also be class dependent.

that are larger in magnitude than some predetermined threshold. This is called *Threshold Coding*. However, in practice the overhead information required to tell the receiver which coefficients were sent and which were not is sometimes excessive. Thus, a more efficient technique, called *Zonal Coding*, is simply to transmit the coefficients in order of increasing spatial frequency or some other predetermined order, and to send an end-of-block code-word when all the remaining coefficients are below threshold.

A simplification of adaptive coding with block classification is to make two passes over each image, and during the first pass measure the spatial activity of each block.[5.3.15] During the second pass each block is classified in such a way that the proportions $\{w_\ell\}$ of blocks in the classes $\{C_\ell\}$ equals some predetermined distribution $\{W_\ell\}$. If each class is then coded using, for example, Lloyd-Max fixed word-length codes, then the bit-rate R_ℓ for each class is constant, and so is the bit-rate R per image. More precisely,

$$R = \sum_{\ell=1}^{L} W_\ell R_\ell \text{ bits/pel} \tag{5.3.106}$$

The distortion is not constant, however, since an excessive number of high-detail blocks will result in some of them being classified into low-detail classes that are coded at a lower bit-rate and thus with higher distortion.

The preceding theoretical framework for transform coding should not be taken to be the final word on the subject. Indeed a great deal of experimental work remains undone, and much remains to be learned. For example, large block-sizes exploit statistical redundancy better than small ones. However, small blocks are better able to utilize subjective redundancy and to adapt to changing statistics within the image. Also, the mean square error distortion criterion treats quantization error and truncation error equally, whereas their subjective effects are quite different. Moreover, at low bit-rates, the edge of the blocks become visible, and this distortion tends to overshadow that of the interior pels. Receiver low-pass spatial filtering can reduce block-edge visibility. However, an approach that is often better in terms of picture quality is to simply filter at the transmitter and reduce the pel sampling rate prior to transform coding.

Intraframe transform coders generally aim for bit-rates below one bit/pel. In order to achieve this with adequate picture quality, the images must contain some proportion of low-detail, flat area blocks that can be coded at very low bit-rate. This then makes additional bits available for coding the high-detail blocks. Since for a given sampling rate DPCM generally lacks the ability to code low-detail areas at extremely low bit-rates, under these circumstances transform coding usually outperforms DPCM.

5.3.4 Transform Coding of Color Pictures

The most straightforward approach to transform coding of color pictures is to code each of the three color components* separately using the monochrome coding techniques described above. Typically, the first step is to convert the original *RGB* components to luminance and chrominance components such as *YIQ* that are reasonably uncorrelated with each other. We saw in Chapter 3 that for a given picture or class of pictures the Karhunen-Loeve techniques can be used to derive a linear transformation of *RGB* into three new components that are *completely* uncorrelated. However, we also saw that there are few benefits for doing so as compared with simply using *YIQ*. For multispectral images or pictures with unusual color distributions this may not be the case.

In this section we will be concerned mainly with coding the chrominance, since the previous sections have already discussed luminance coding. We would expect that the bit-rates required for the I and Q components, as compared with the luminance component Y, would be approximately proportional to their relative bandwidths. For example, corresponding to NTSC conventions the required I and Q signal bandwidths are approximately 0.36 and 0.12, respectively, of the Y signal bandwidth. Thus, a two-dimensional transform coder, which reduces bandwidth in both dimensions, should be able to code the I and Q components using about $(0.36)^2 + (0.12)^2 \approx 14\%$ of the bit-rate of the Y component. Moreover, since the chrominance signal power is significantly smaller than that of the luminance, we would expect the chrominance bit-rate to be even lower than this estimate.

Thus, as with PCM and DPCM color image coding, the first step in coding the I and Q signals should be some sort of bandlimiting operation. It could be done prior to transform coding by the use of digital and analog filters, and in this case the sampling rate could also be reduced. Alternatively, if the transform block size is large enough, bandlimiting may be effected by simply discarding high-frequency transform coefficients. However, if the block size is small, such action may lead to undersirable visibility of distortion at edges of the transform blocks.

After bandlimiting, the chrominance signals may be transform coded using the nonadaptive or adaptive techniques discussed earlier for monochrome signals. However, a complication arises in determining the relative proportion of bits to be allocated to color components. This problem has yet to be solved completely. However, a few approaches can be suggested as starting points. For example, given the luminance distortion D_Y, we may be able to uniquely

---

* Sometimes more than three parameters are used, e.g., multispectral satellite images (see Chapter 2).

specify the corresponding chrominance distortions. If I and Q have not yet been bandlimited and p_Y is the percentage of luminance 2D-transform coefficients sent, then we might choose

$$p_I = (0.36)^2 p_Y$$

$$p_Q = (0.12)^2 p_Y \tag{5.3.107}$$

which corresponds to the NTSC relative bandwidths. These would, in turn, determine the chrominance bit-rates and distortions, as for example in Eqs. (5.3.78-83). Alternatively, we may be able to define subjective chrominance spatial frequency weightings such that the overall image distortion is a sum of that in the components, i.e.,

$$D = D_Y + D_I + D_Q \tag{5.3.108}$$

The problem then reduces to one of simply coding $3N$ transform coefficients with minimum distortion or bit-rate, in which case the techniques for luminance transform coding described in previous sections may be used. For example, suppose the $3N$ transform coefficients are interleaved and reordered according to decreasing frequency weighted MSV, i.e., from Eq. (5.3.81)

$$\theta_m \triangleq \frac{1.68\sigma_m^2}{A_F\, T_m} \quad 1 \le m \le 3N \tag{5.3.109}$$

is monotonically nonincreasing. Then, for uniform quantization and entropy coding we have from Eqs. (5.3.78-83)

$$R = \frac{1}{1.39(3N)} \sum_{m=1}^{3pN} \ln \frac{1.68\ \sigma_m^2}{\theta_{3pN}\, A_F\, T_m} \text{ bits/pel}$$

$$D = \frac{p\theta_{3pN}}{1.39} + \frac{1}{N} \sum_{m>3pN}^{3N} \frac{\sigma_m^2}{A_F\, T_m} \tag{5.3.110}$$

where $p < 1$ is the fraction of the $3N$ coefficients that are transmitted.

In reality, as we have seen in Chapter 4, optimization of luminance/chrominance distortion is a very complex nonlinear problem that depends on many subjective relationships between luminance, chromaticity, spatial detail, etc. Many of the statistical and subjective phenomena described

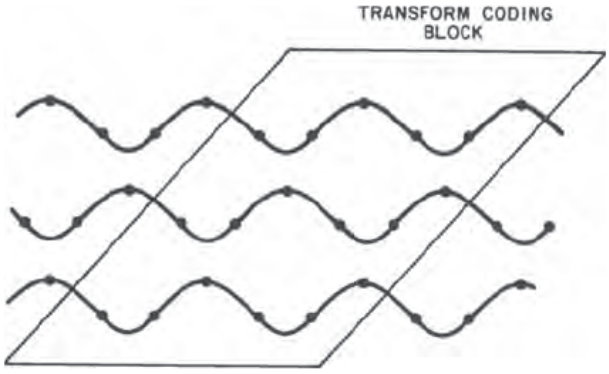


Fig. 5.3.28 Successive lines of an NTSC color waveform sampled at three times the color subcarrier frequency. If two-dimensional transform blocks are chosen appropriately, then line-to-line correlation between pels is high.

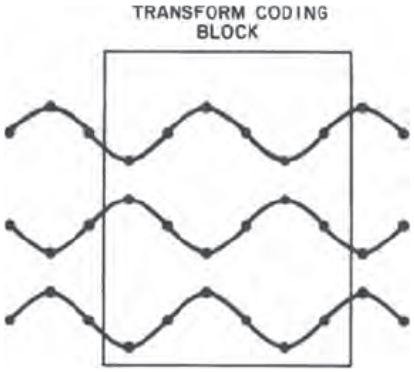


Fig. 5.3.29 Successive lines of an NTSC color waveform sampled at four times the color subcarrier frequency. Vertically adjacent pels are anticorrelated.

earlier for DPCM coding of color images may also be exploited for transform coding. For example, if a transform block has a high-detail luminance component, then it also usually has high-detail chrominance components. A high chrominance distortion may then be allowed not only because of self-masking by the chrominance, but also because of masking by the high-activity luminance component. Unlike DPCM however, the distortion is then spread throughout the block, which in the case of large block-sizes may limit the extent to which such masking may be exploited without undesirable degradation.

Transform coding has also been applied to composite color waveforms such as the NTSC and PAL color signals. In this application it is important that some of the basis vectors be chosen specifically to represent the amplitude and phase of the color subcarrier signal. This requires that the original sampling rate be synchronized in some way to the color subcarrier and that the block size be carefully chosen. It also requires both sine and cosine color subcarrier basis vectors, at least in the horizontal direction. These are present in the RDFT and KLT, but not in the DCT and WHT. In principle, a new set of orthonormal basis vectors could be constructed* from the two color subcarrier vectors plus most of the DCT vectors. However, this has not been studied in detail.

Line-to-line vertical correlation in the sampled composite color signal depends on the sampling rate, the block shape and the format of the composite waveform. For example, Fig. 5.3.28 shows an NTSC signal sampled at three times color subcarrier frequency. If the block boundary is chosen as shown, then the result is a high vertical correlation that can be exploited by a 2D transform. Fig. 5,3.29 shows an NTSC signal sampled at four times color subcarrier frequency. Because the scan line duration is 227.5 color subcarrier periods, vertically adjacent pels are anticorrelated. This behavior is exploited by the basis vector

$$t' = (1 \ -1 \ \ 1 \ -1 \ \ 1 \ ...) \tag{5.3.111}$$

which is already present in most of the transforms discussed so far.

The PAL composite signal has a color subcarrier phase shift at the end of each scan line. This reduces the amount of vertical correlation when periodic sampling is employed. However, if the phase of the sampling is also shifted at the end of each scan line, high vertical correlation can be restored. The SECAM composite signal has different color subcarrier frequencies on adjacent scan lines, and therefore vertical correlation is lower than with NTSC or PAL.

---

* Using, for example, the Gram-Schmidt orthogonalization procedure.[5.3.16]

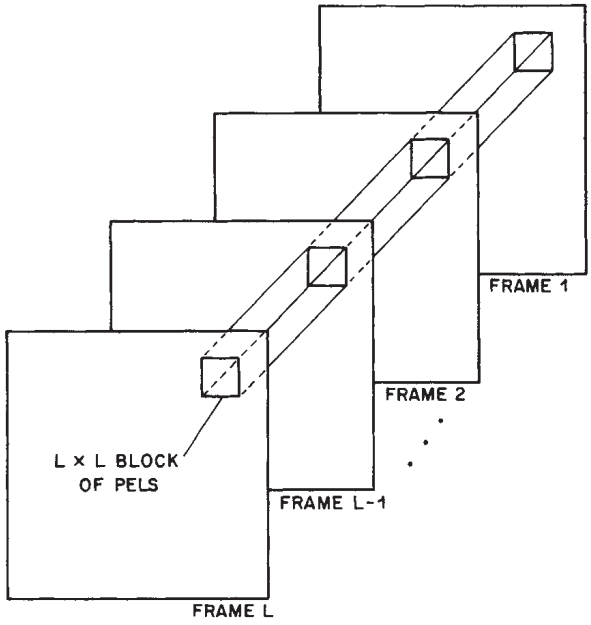


Fig. 5.3.30 Three-dimensional transform coding may be applied to $L \times L \times L$ blocks of pels taken from L successive frames of imagery.

Experimental results so far on intraframe transform coding of composite signals are mixed.[5.3.17] Picture quality versus bit-rate seems to be comparable to that which can be achieved with DPCM coding of composite waveforms. Sensitivity to transmission bit errors may be sometimes lower than DPCM. However, complexity and therefore cost are considerably higher.

As with DPCM, transform coding of the composite signal is somewhat simpler than component coding. However, the overall performance is not as good in terms of bit-rate and picture quality. Moreover, as with DPCM, component coding enjoys a considerable advantage in being relatively compatible with NTSC, PAL and SECAM color signals, once the composite-to-component conversion has been made. This aspect is very important in international broadcasting of color TV signals.

5.3.5 Interframe or Three-Dimensional Transform Coding

With multiple frame imagery, such as motion picture film or television, a logical extension of intraframe coding concepts is three-dimensional transform coding. With this approach the frames are coded L at a time, and the data is partitioned into $L \times L \times L$ blocks† of pels prior to transformation as shown in Fig. 5.3.30. Each block could then be arranged into a column vector of length $N = L^3$ and transformed via an Nth order transform. However, as before in the case of 2D-transforms, the usual approach is to use a *separable* 3D-transform, i.e., three successive applications of the L-th order transform to the horizontal, vertical and temporal axes of each pel block. Thus, if $\boldsymbol{B} = [b_{ijk}]$ is the three dimensional block of pels to be transformed, then the $L \times L \times L$ block of transform coefficients is $\boldsymbol{C} = [c_{\ell mn}]$, where

$$c_{\ell mn} = \sum_{i,j,k=1}^{L} b_{ij\,k}\; t_{\ell i}\; t_{mj}\; t_{nk} \tag{5.3.112}$$

and the t's are elements of the $L \times L$ transform matrix $\boldsymbol{T}$. If i, j and k are the vertical, horizontal and temporal coordinates, respectively, then ℓ and m are the vertical and horizontal spatial frequency coordinates, and n is the temporal frequency coordinate.

As with intraframe transforms, the interframe transform coefficients usually have Laplacian PD's, except for the DC coefficient, which is generally considered to be uniformly distributed.[5.3.18] However, interframe transform coding is affected not only by the differing amounts of spatial detail within a frame, but also by variations in the amount of movement and other temporal

† Generalization to non-cubic blocks is straightforward.

activity in different regions of the picture. Thus, there are two sources of statistical nonstationarity at work, and therefore it is usually *mandatory* that adaptive techniques be used in order to achieve good results. In fact, experimental evidence[5.3.19] indicates that nonadaptive interframe transform coding of 16×16×16 pel blocks performs only slightly better than nonadaptive intraframe coding of 16×16 pel blocks, which is much less costly, by far, to implement.

If a region of the picture has low spatial detail then, as with intraframe transform coding, only the coefficients having low spatial frequency need be transmitted. Correspondingly, if a region has low or zero frame-to-frame changes then only the coefficients having low or zero temporal frequency, need be transmitted.

The problem of quantifying temporal activity for each three-dimensional block of pels and of adapting the coding of coefficients thereto is directly analogous to that of two-dimensional adaptive transform coding where spatial activity is the controlling parameter. However, due to the temporal averaging property of most TV cameras, it is very rare that a 3D-transform block will have simultaneously high spatial and high temporal activity. Thus, the non-DC coefficient energy is not a good activity measure for coding purposes. Instead, a selected set of spatial frequency coefficients [$n=0$ in Eq. (5.3.112)] should be used to indicate spatial activity, and a selected set of temporal frequency coefficients [$\ell=m=0$] should be used to indicate temporal activity.[5.3.20] Classification techniques may then be used that categorize each block according to the amount of spatial and/or temporal activity and, as in adaptive 2D-transform coding, code using a coefficient bit-assignment and quantization based on the subjective visibility of distortion in each class. Adaptive interframe coding experiments have yielded a halving of the bit-rate compared with nonadaptive coding.[5.3.19]

While, in principle, all of the above techniques will contribute to bit-rate reduction, in practice there are many pitfalls. For example, the frequency weighted mean square error distortion criterion is not well understood even for intraframe coding, and is even less understood as a measure of interframe coding artifacts and temporal resolution. In real-time, constant frame-rate applications the cost of implementing a three-dimensional transform coder can be prohibitively high compared with other coding techniques that may be equally or only slightly less efficient. For a block size of $L\times L\times L$ pels, L frames of memory are needed for the basic coding operation. In addition, some input buffer memory is required as well as the usual output data buffer associated with any adaptive-coding/constant-channel-rate system, and this causes additional transmission delay. With these and other impediments, it is not surprising that the technology of interframe block transform coding has been slow to develop.

5.4 Hybrid Transform Coding Techniques

We have seen from the previous sections that transform coding involves high complexity, both in terms of storage and computational requirements. For sources with stationary statistics, transform coding with large block size removes essentially all of the statistical redundancy present in the source signal. However, for real-world picture sources that do not have stationary statistics, block sizes should be smaller so that adaptive techniques can accommodate to the changes in local area statistics. Small block sizes should also be used if the adaptive coding is to take advantage of psychovisual properties of the human observer. However, with smaller block sizes, considerable block-to-block redundancy will exist even after transform coding. It is this dilemma that is partially alleviated by Hybrid Transform Coding. As we shall see, storage requirements and computational complexity are also reduced by hybrid transform coding.

5.4.1 Intraframe Hybrid Transform Coding

Basically, hybrid transform coding combines *transform* coding and *predictive* coding in order to reduce storage requirements and to increase adaptability and efficiency. Intraframe hybrid transform coding is typically implemented by first transform coding in one spatial dimension and then predictive (DPCM) coding in the other spatial dimension.[5.4.1]

For example, suppose each image to be coded is partitioned into vertical strips, L-pels wide as shown in Fig. 5.4.1. Let $\boldsymbol{B} = [b_{ij}]$ be such a strip. We then perform an Lth order 1D-transform on each row of $\boldsymbol{B}$ to obtain the coefficients

$$c_{im} = \sum_{j=1}^{L} b_{ij}\, t_{mj} \tag{5.4.1}$$

where the t's are the elements of the $L \times L$ transform matrix $\boldsymbol{T}$. This operation is a transformation in the horizontal (row) direction and serves to remove horizontal redundancy between pels that are in the same row of $\boldsymbol{B}$. We denote the vector of transform coefficients of row i, strip k by $\boldsymbol{c}_i^k$.

With intraframe hybrid transform coding we now remove vertical redundancy in $\boldsymbol{B}$ by employing DPCM in the vertical direction to code the transform coefficients. For example, with single element predictive coding, we would quantize and code the differential vector

$$\boldsymbol{e}_i^k \triangleq \boldsymbol{c}_i^k - \tilde{\boldsymbol{c}}_{i-1}^k \tag{5.4.2}$$

where $\tilde{\boldsymbol{c}}_{i-1}^k$ is the quantized version of the previous row transform coefficients in strip k. In the generalized case, all of the previously transmitted coefficients

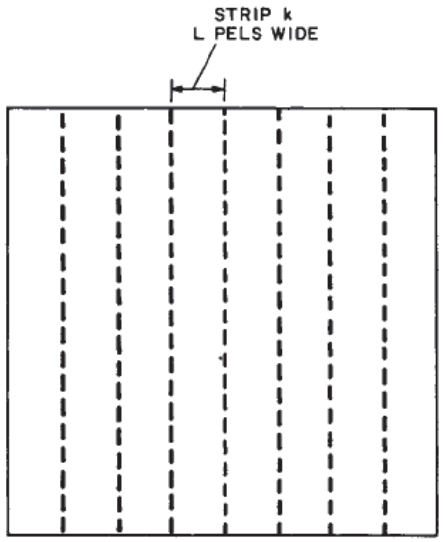


Fig. 5.4.1 With intraframe hybrid coding the image is first partitioned into vertical strips, each having a width of L pels.

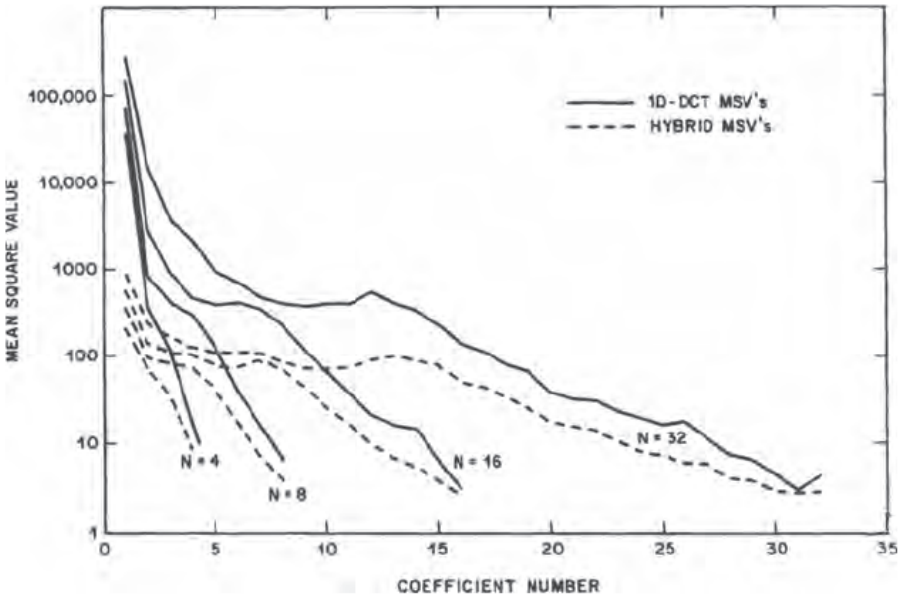


Fig. 5.4.2 Comparison of 1D-DCT coefficient MSV's with 2D hybrid MSV's for block lengths 4, 8, 16, 32. In all cases the hybrid MSV's are smaller. However, the difference is small for the higher order coefficients.

$$\tilde{c}_q^{\ell} = \begin{cases} q < i, \ \ell = \text{any} \\ q = i, \ \ell < k \end{cases} \tag{5.4.3}$$

could be used to compute a prediction vector $\hat{c}_i^k$ for c_i^k, and the prediction error vector

$$e_i^k = c_i^k - \hat{c}_i^k \tag{5.4.4}$$

would then be quantized, coded and transmitted. We denote the quantized version by $\tilde{e}_i^k$.

In the absence of transmission errors, the receiver can compute $\hat{c}_i^k$ from the previously received information and reconstruct the block of pels as follows:

$$\tilde{c}_i^k = \hat{c}_i^k + \tilde{e}_i^k$$

$$\tilde{b}_i^k = T^{-1} \tilde{c}_i^k \tag{5.4.5}$$

assuming vectors are arranged in column matrices.

Hybrid intraframe transform coding with line-to-line DPCM requires only one line of memory instead of the L lines required of an $L \times L$, 2D transform coding. Moreover, with line-to-line processing the capability of adapting to local variations in the image is significantly enhanced compared with 2D transform coding.

Fig. 5.4.2 shows DCT hybrid transform statistics for the image "Karen" using various one-dimensional block sizes and simple line-to-line differences of transform coefficients. MSV's for both the differential signals and the respective one-dimensional transform coefficients are shown. The differential signal probability distributions all have the characteristic Laplacian shape, and for the lower order coefficients their MSV's are significantly smaller than the corresponding one-dimensional DCT coefficients. However, for the higher order coefficients the difference is not so great. In fact, for images containing high detail, the higher order coefficients have little or no correlation from line to line. In this case, the best predictor for them is zero, i.e., they should be coded independently of previously transmitted values.[5.4.3]

The theoretical framework for quantization and coding of hybrid transform values is very similar to that of transform coding itself. For example, consider a single strip $B = [b_{ij}]$ and simple line differential PCM, i.e.,

$$\hat{c}_i = \tilde{c}_{i-1} \tag{5.4.6}$$

Also, as in previous sections, suppose we utilize the frequency weighted mean square error distortion criterion*

$$D = \frac{1}{L} \sum_{m=1}^{L} \frac{E(c_{im} - \tilde{c}_{im})^2}{A_F\, T_m}$$

$$= \frac{1}{L} \sum_{m=1}^{L} \frac{E(e_{im} - \tilde{e}_{im})^2}{A_F\, T_m} \tag{5.4.7}$$

and that the e's are quantized and coded independently of each other. In Section 5.3.2 we saw that under these circumstances, the bit-rate depends only on the MSV's (arranged in decreasing order) of the quantities to be transmitted, i.e.,

$$\begin{aligned} \sigma_{im}^2 &= E(e_{im}^2) \\ &= E(c_{im} - c_{i-1,m})^2 \\ &= E\left[\sum_{j=1}^{L} (b_{ij} - b_{i-1,j})\, t_{mj}\right]^2 \end{aligned} \tag{5.4.8}$$

where we assume that $\tilde{c}_{i-1,m} \approx c_{i-1,m}$. Thus,

$$\sigma_{im}^2 = E\left[\sum_{j=1}^{L} (b_{ij} - b_{i-1,j})\, t_{mj} \sum_{k=1}^{L} (b_{ik} - b_{i-1,k})\, t_{mk}\right]$$

$$= \sum_{j,k=1}^{L} \left\{r_{i\,j\,i\,k} - r_{i,j,i-1,k} - r_{i-1,j,i,k} + r_{i-1,j,i-1,k}\right\} t_{mj}\, t_{mk}$$

* As before, in some cases the statistical averaging operator E may be profitably replaced by a subjectively weighted averaging.

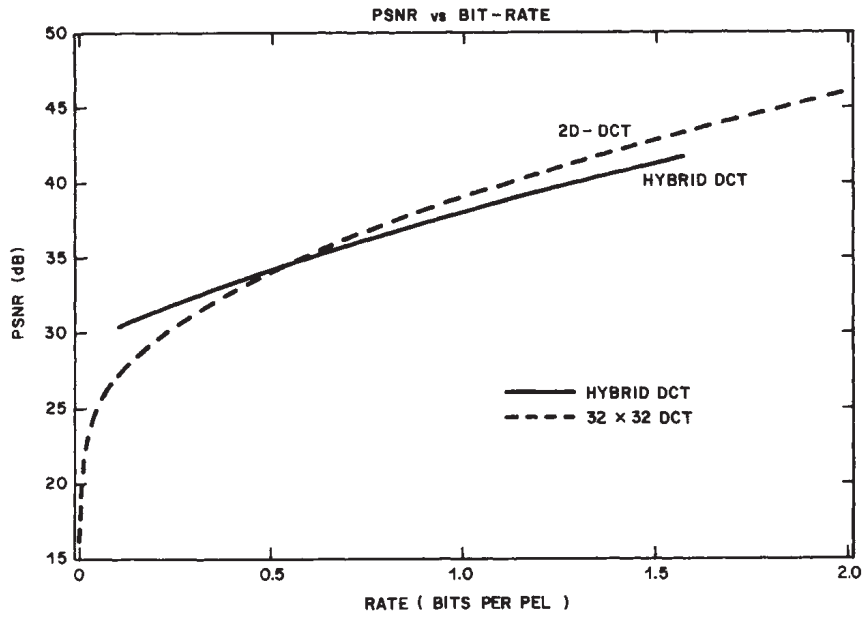


Fig. 5.4.3 Comparison of hybrid coding (DCT, L=32) with two-dimensional transform coding (separable DCT, 32×32 blocks) using the image "Karen". Both cases employed uniform quantization with entropy coding.

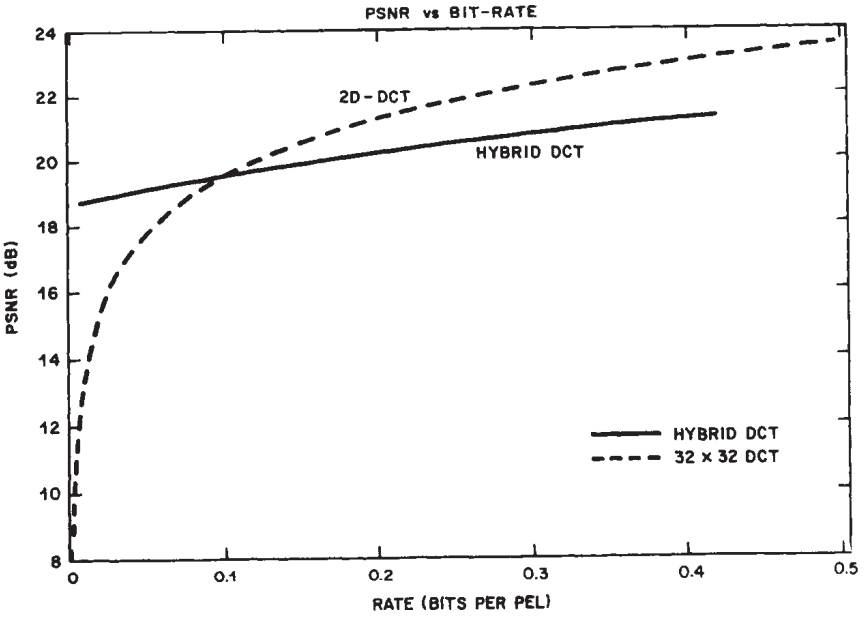


Fig. 5.4.4 Comparison of hybrid coding and transform coding using the isotropic correlation model with $\rho = 0.98$. The same coding as Fig. 5.4.3 was used.

where the correlations are given by

$$r_{i\,j\,k\ell} \triangleq E\,(b_{ij}\,b_{k\ell}) \tag{5.4.9}$$

For example, with uniform quantization and entropy coding the bit assignments and distortion are given by Eqs. (5.3.78-83), and the quantizer step size is given by Eq. (5.3.85). In particular, if the block-size is not too small and the fraction of transmitted coefficients is $p < 1$, then

$$\theta \approx \frac{1.68\sigma_{pL}^2}{A_F T_{pL}}$$

and

$$R \approx \frac{1}{1.39L} \sum_{m=1}^{pL} \ln \frac{\sigma_m^2\, T_{pL}}{T_m\, \sigma_{pL}^2} \quad \text{bits/pel}$$

$$D \approx \frac{1.21p\ \sigma_{pL}^2}{A_F\ T_{pL}} + \frac{1}{L} \sum_{m>pL}^{L} \frac{\sigma_m^2}{A_F\ T_m} \tag{5.4.10}$$

Fig. 5.4.3 shows *PSNR* vs R for "Karen" using $A_F\ T_m = 1$ and a block length $L = 32$ pels. Fig. 5.4.4 shows similar results for the isotropic correlation model

$$r_{i\,j\,k\ell} = \frac{255^2}{3}\,\rho^{\sqrt{(i-k)^2 + (j-\ell)^2}} \tag{5.4.11}$$

Also shown for reference in these figures are the results of the separable 2D-DCT coding using 32×32 pel blocks. We see that hybrid and 2D transform coding have comparable performance, at least for the MSE distortion criterion.

Fig. 5.4.5 shows a generalized block diagram for intraframe hybrid transform coding. The one-dimensional transform converts $1\times L$ blocks of pels $\boldsymbol{b}_i^k$ into blocks of coefficients $\boldsymbol{c}_i^k$ that are then DPCM coded. The predictor uses one or more previously transmitted coefficient blocks to compute the prediction vector $\hat{\boldsymbol{c}}_i^k$. In fact, previously transmitted coefficients in the *present* block could also be used to predict each coefficient. The differential vector $\boldsymbol{e}_i^k$ is then quantized, coded and transmitted to the receiver where DPCM decoding and inverse transformation take place in order to produce the pel vector $\tilde{\boldsymbol{b}}_i^k$.

The processor of Fig. 5.4.5 has all the elements necessary to carry out the adaptive coding techniques described in Section 5.3. In particular, the quantizer can be adapted to line-to-line variations in the picture, and coefficients can be discarded or quantized according to whatever quality criterion is used. If the

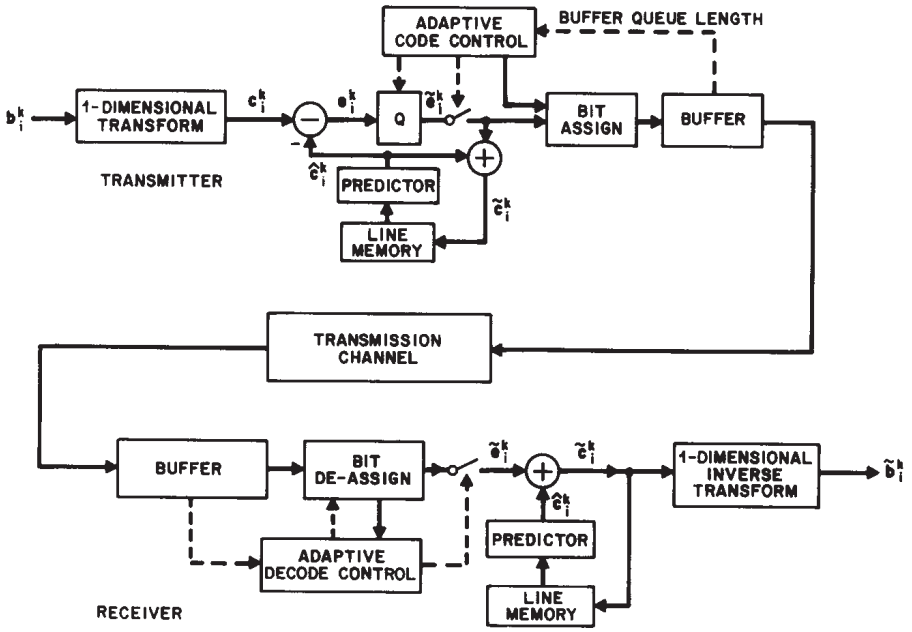


Fig. 5.4.5 Generalized intraframe hybrid coder. Line i of strip k is 1D-transformed into c_i^k, which is then DPCM coded into a quantized differential vector $\tilde{e}_i^k$. The inverse process takes place at the receiver.

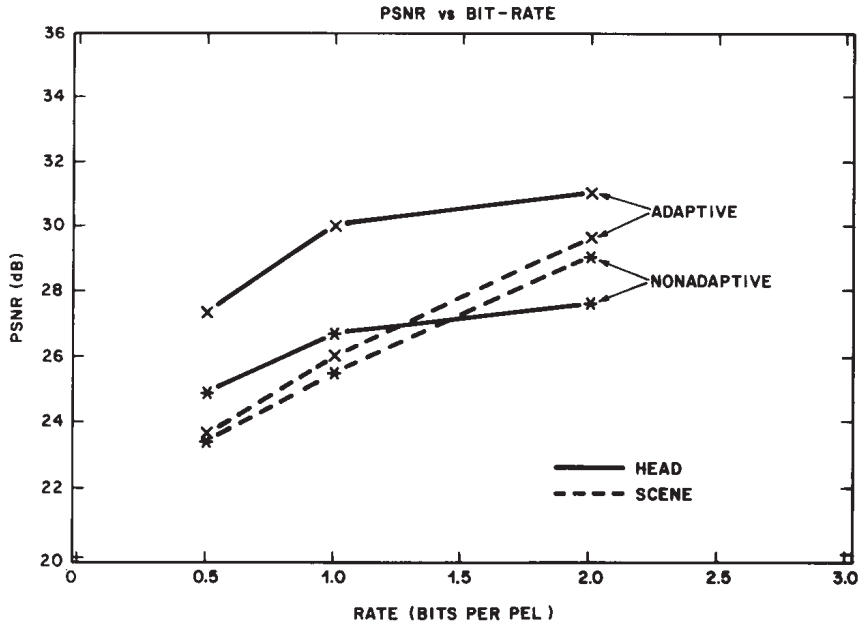


Fig. 5.4.6 Experimental results using intraframe hybrid coding on the "Head" and "Scene" images of Fig. 5.4.7. Data points * are for nonadaptive coding. Points × are for adaptive coding.

Fig. 5.4.7 Images used for hybrid coding studies. (top) Head. (bottom) Scene.

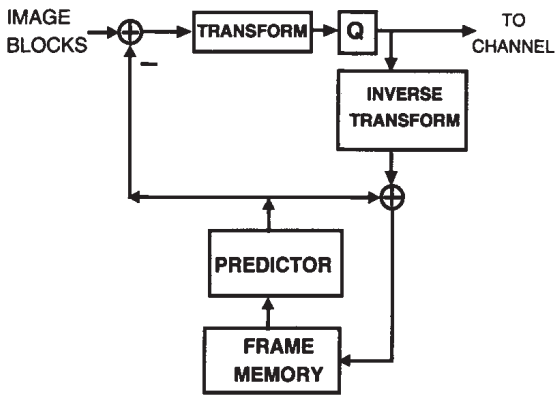


Fig. 5.4.8a Interframe hybrid transform coder for pel domain predictors. Here the prediction error is transformed instead the pels themselves.

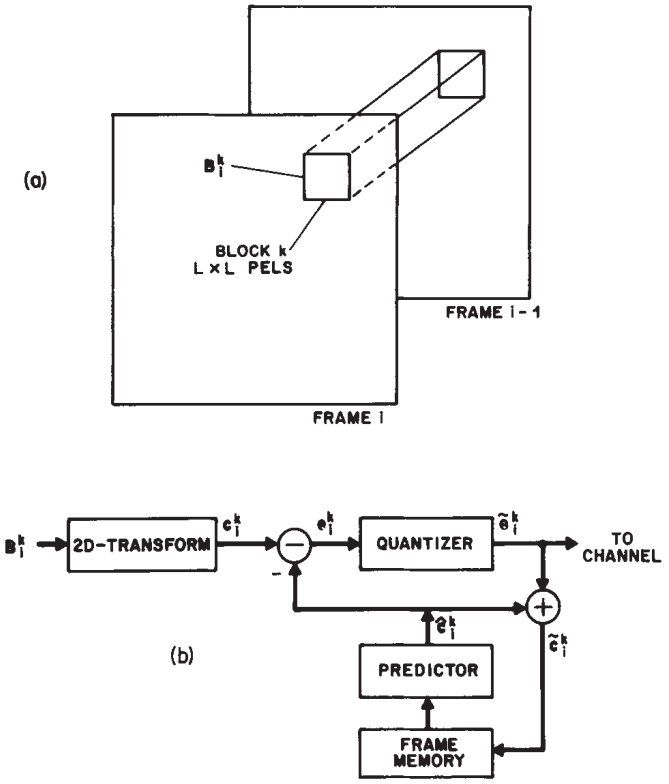


Fig. 5.4.8 Interframe hybrid coding. Blocks of $L \times L$ pels are first 2D-transformed. The resulting coefficients are then interframe DPCM coded and sent to the receiver where the inverse process takes place.

number of transmitted bits varies greatly from block-to-block and if the transmission channel operates at a constant bit-rate, then the buffers shown in Fig. 5.4.5 may have to be quite large in order to achieve a relatively low channel bit-rate. In this case the transmitter adaptive coding must, in addition to its other constraints, make sure that the transmitter buffer neither empties nor overflows. The receiver adaptive decoding must take into account the possibility of transmission bit errors, and in the event they occur the receiver must recover as fast as possible to avoid data being lost due to receiver buffer overflow.

The transmitter adaptive code controller will typically use some measure of spatial activity as well as the buffer queue length in order to determine the quantization accuracy and the number of coefficients that must be sent in a particular block. This controller may also modify the bit assignment algorithm, e.g., a Huffman or run-length code, in order to adapt to changing statistics of the transform coefficients and differential values. Such coding strategy overhead may have to be transmitted to the receiver, or in some cases it may be derived from the coded information itself.

Fig. 5.4.6 shows PSNR vs R results for adaptive intraframe hybrid transform coding of the two images shown in Fig. 5.4.7. One is fairly low-detail and the other is fairly high-detail.[5.4.2] In this coder the hybrid DCT was used with strip width $L = 32$ pels. In each strip the MSV's of the differential values were measured and used to design Lloyd-Max quantizers. Thus, adaptation to changing image statistics was only in the horizontal direction. Significant improvement over nonadaptive coding resulted only for the head and shoulders image, where detail was nonuniformly distributed in the picture.

5.4.2 Interframe Hybrid Transform Coding

Three dimensional $L \times L \times L$ block transform coding typically requires memory at the transmitter to store L frames of image data. A similar requirement exists at the receiver. This need can be reduced to approximately one frame of memory at the transmitter and one at the receiver by the use of interframe hybrid transform coding.

The first step is to partition each image frame into two-dimensional blocks of $L \times L$ pels. We represent the kth block of pels in the ith frame by $\boldsymbol{B}_i^k$. Fig. 5.4.8 shows a straightforward extension of intraframe hybrid coding in which each block is transformed by a 2D-transform producing a length L^2 vector of coefficients $\boldsymbol{c}_i^k$. As in the previous section, a predictor uses previously transmitted blocks of coefficients in order to compute a prediction vector $\hat{\boldsymbol{c}}_i^k$ that, in the absence of transmission errors, is available both at the transmitter and receiver. However, in this case the predictor has access to the previous frame blocks. For example, a single-block previous-frame prediction would produce

$$\boldsymbol{e}_i^k = \boldsymbol{c}_i^k - \tilde{\boldsymbol{c}}_{i-1}^k \tag{5.4.12}$$

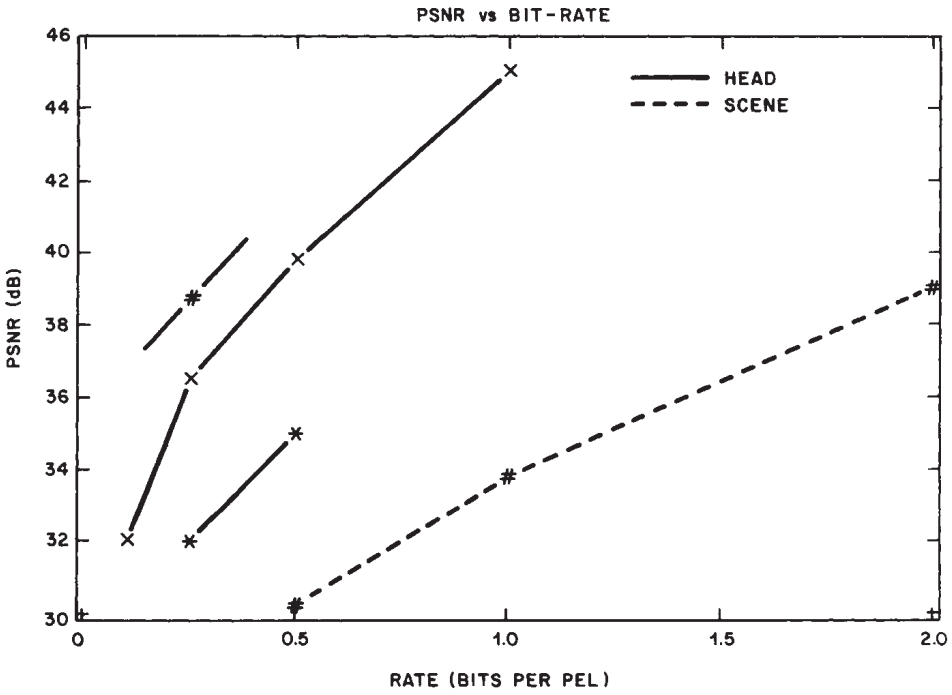


Fig. 5.4.9 Experimental results using interframe hybrid coding on the images of Fig. 5.4.7. Data points * are for nonadaptive coding, points × are for adaptive coding and points # are for adaptive coding with movement compensation.

Fig. 5.4.8a shows an equivalent interframe hybrid transform coding system, except that here the predictor operates in the pel domain instead of the transform domain. This arrangement is particularly useful when the prediction operation includes motion compensation, since motion estimation is difficult to perform in the transform domain.

Interframe hybrid transform coding differs significantly from its intraframe counterpart in that the non DC elements of the vector $\boldsymbol{c}_i^k$ are determined exclusively by the amount of *spatial* detail in the picture, whereas the DPCM differential transform vector $\boldsymbol{e}_i^k$ depends both on the amount of spatial detail and on the amount of frame-to-frame change, i.e., *temporal* activity in the scene. Thus, in regions of the picture containing low detail, many of the elements of $\boldsymbol{c}_i^k$ will be negligible, which in turn causes many of the elements of $\boldsymbol{e}_i^k$ to be negligible. In regions of the picture containing no movement, all elements of $\boldsymbol{e}_i^k$ should be negligible regardless of the amount of detail. However, camera noise and instabilities will often cause frame-to-frame differences even in the absence of movement. As with intraframe transform coding, adaptive coding is essential in order to achieve the greatest benefits of interframe hybrid transform coding. With interframe transform coding, nonadaptive operation is only slightly better than intraframe transform coding.[5.3.19]

With television, interlace is usually employed, with each frame consisting of two line-interleaved fields that are transmitted one after the other. Thus, if $\boldsymbol{B}_i^k$ represents the kth block of pels in the ith *field,* then for true frame differential operation we should define

$$\boldsymbol{e}_i^k = \boldsymbol{c}_i^k - \tilde{\boldsymbol{c}}_{i-2}^k \tag{5.4.13}$$

However, using the definition of Eq. (5.4.12) should not be too detrimental, except that highly detailed picture areas containing no movement will generate nonnegligible field differential vectors $\boldsymbol{e}_i^k$, whereas with the frame-differential of Eq. (5.4.13) this does not happen.

Adaptive interframe hybrid transform coding generally follows the same classification strategy described earlier for interframe transform coding. However, in this case temporal activity is measured by frame-to-frame coefficient or pel differences instead of transform coefficient MSV's. Experimental results of adaptive interframe hybrid transform coding are comparable to adaptive interframe nonhybrid transform coding,[5.3.19] which is much more complex and therefore more expensive to implement. Adaptation can reduce the required bit-rate by factors of 2 to 10, depending on the algorithm and the amount of spatial and temporal activity in the scene.

Fig. 5.4.9 shows *PSNR* vs R results for adaptive interframe hybrid transform coding of the images in Fig. 5.4.7. The separable 2D-DCT was used with block size 16×16. In each block the MSV's of the frame differential values were measured and used to design Lloyd-Max quantizers. As seen in the figure,

adaptive interframe coding shows significant improvement over nonadaptive coding. Interframe hybrid coding will be discussed in more detail in Chapter 6.

With interframe hybrid transform coding, motion compensation can also be used to reduce the bit-rate significantly. The most straightforward approach is to first estimate the frame-to-frame translation for the block of pels to be coded using techniques for block matching motion estimation that were covered in Section 5.2.3. Then the block of pels to be used as a prediction in the previous frame is displaced by the measured amount, followed possibly by a recalculation of transform coefficients prior to subtraction from the incoming data. Alternatively, displaced frame differences could be computed prior to transformation, using techniques described earlier for DPCM. The advantage of sending transforms of differential signals instead of the differential signals themselves lies in the greater ability of transforms to send large areas of low spatial-frequency signals.

Experimental results[5.3.19] indicate that motion compensation can reduce the bit-rate by up to a factor of two compared with simple interframe hybrid transform coding (see Fig. 5.4.9). However, results are extremely dependent on the type of picture and the amount of motion. More will be said about motion compensated hybrid transform coding in later Chapters.

5.4.3 Interrelation Between Hybrid Transform Coding and Predictive Coding

Analytical results for hybrid transform coding, as with other areas of image analysis, rely heavily on the assumption of statistical stationarity and on the mean square error distortion criterion. While in many situations this does not reflect reality, the analysis may be useful in some aspects of adaptive coder design, particularly in the coding of classes having relatively constant spatial detail or temporal activity.

We begin by assuming that blocks of pels are rearranged into column vectors and that their statistics are wide-sense stationary, i.e., the correlation† matrix of the pel vectors

$$\boldsymbol{R}_b \triangleq E(\boldsymbol{b}\boldsymbol{b}') \tag{5.4.14}$$

is constant and well defined. In this case the correlation matrix of the transform coefficients is easily shown to be

† Prime indicates conjugate transpose.

$$\boldsymbol{R}_c \triangleq E(\boldsymbol{c}\boldsymbol{c}') = \boldsymbol{T}\boldsymbol{R}_b\boldsymbol{T}' \tag{5.4.15}$$

The diagonal elements of $\boldsymbol{R}_c$ are then the variances of the transform coefficients.

We now denote by $\dot{\boldsymbol{c}} = \boldsymbol{T}\dot{\boldsymbol{b}}$ the already quantized and transmitted block of coefficients in the previous line, field or frame that is to be used in the prediction of $\boldsymbol{c}$. The cross correlation matrix between $\boldsymbol{c}$ and $\dot{\boldsymbol{c}}$ is then given by

$$\boldsymbol{R}_{c\dot{c}} \triangleq E(\boldsymbol{c}\dot{\boldsymbol{c}}') = \boldsymbol{T}\,\boldsymbol{R}_{b\dot{b}}\,\boldsymbol{T}' \tag{5.4.16}$$

where

$$\boldsymbol{R}_{b\dot{b}} \triangleq E(\boldsymbol{b}\dot{\boldsymbol{b}}') \tag{5.4.17}$$

is the corresponding cross correlation between $\boldsymbol{b}$ and the previously transmitted block $\dot{\boldsymbol{b}}$. If the effects of quantization are small, i.e., the received picture quality is high, then $\boldsymbol{R}_{b\dot{b}}$ can be found either by direct measurement or from the assumed statistical model of the original pels.

The minimum mean square error (MSE) linear prediction of $\boldsymbol{c}$ given $\dot{\boldsymbol{c}}$ is then found by elementary calculus (see also Sec. 3.1.5) to be

$$\hat{\boldsymbol{c}} = \boldsymbol{A}\dot{\boldsymbol{c}} \tag{5.4.18}$$

where

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{R}_{c\dot{c}}\,\boldsymbol{R}_c^{-1} \\ &= \boldsymbol{T}\,\boldsymbol{R}_{b\dot{b}}\,\boldsymbol{R}_b^{-1}\,\boldsymbol{T}' \end{aligned} \tag{5.4.19}$$

Similarly, we can show that

$$\hat{\boldsymbol{b}} \triangleq \boldsymbol{R}_{b\dot{b}}\,\boldsymbol{R}_b^{-1}\,\dot{\boldsymbol{b}} \tag{5.4.20}$$

is the minimum MSE linear predictor of $\boldsymbol{b}$ given $\dot{\boldsymbol{b}}$. Thus, Eq. (5.4.18) is equivalent to simply taking the transform of $\hat{\boldsymbol{b}}$. Indeed, more complicated predictors that use more than one previous block can be easily derived by simply taking the transform of the corresponding linear or other predictor of the pel vector $\boldsymbol{b}$. For example, suppose we use J previously transmitted blocks and derive a minimum MSE linear predictor for vector $\boldsymbol{b}$. Then we have

$$\hat{\boldsymbol{b}} = \sum_{j=1}^{J} \boldsymbol{A}_j \, \dot{\boldsymbol{b}}_j \tag{5.4.21}$$

where the $\boldsymbol{A}_j$ are found from the statistical model using the techniques of Section 3.1.6. The corresponding transform coefficient predictor is then simply

$$\hat{\boldsymbol{c}} = \sum_{j=1}^{J} \boldsymbol{T} \, \boldsymbol{A}_j \, \boldsymbol{T}' \, \dot{\boldsymbol{c}}_j \tag{5.4.22}$$

Note that if the transform produces truly uncorrelated coefficients then $\boldsymbol{R}_{c\dot{c}}$ and $\boldsymbol{R}_c$ are diagonal matrices, which in turn makes $\boldsymbol{A}$ in Eq. (5.4.19) a diagonal matrix. In this case the single-block minimum MSE linear predictor of the mth element of $\boldsymbol{c}$ is given by

$$\hat{c}_m = \frac{E(c_m \dot{c}'_m)}{E(c_m c'_m)} \, \dot{c}_m \tag{5.4.23}$$

In any event, for the single block predictor of Eq. (5.4.18) the element MSV's of the differential vector $\boldsymbol{e} = \boldsymbol{c} - \hat{\boldsymbol{c}}$ are the diagonal elements of the matrix

$$\begin{aligned} \boldsymbol{V} &= E\{[\boldsymbol{c} - \boldsymbol{A}\dot{\boldsymbol{c}}]\,[\boldsymbol{c} - \boldsymbol{A}\dot{\boldsymbol{c}}]'\} \\ &= \boldsymbol{R}_c - \boldsymbol{R}_{c\dot{c}} \, \boldsymbol{A}' - \boldsymbol{A}\boldsymbol{R}'_{c\dot{c}} + \boldsymbol{A}\boldsymbol{R}_c\boldsymbol{A}' \end{aligned} \tag{5.4.24}$$

If we choose $\boldsymbol{A}$ to be the identity matrix, i.e., use $\dot{\boldsymbol{c}}$ as the prediction, then

$$\boldsymbol{V} = \boldsymbol{T}\left[2\boldsymbol{R}_b - \boldsymbol{R}_{b\dot{b}} - \boldsymbol{R}'_{b\dot{b}}\right] \boldsymbol{T}' \tag{5.4.25}$$

whereas if we choose the optimum single block $\boldsymbol{A}$ of Eq. (5.4.19) then

$$\boldsymbol{V} = \boldsymbol{T}\left[\boldsymbol{R}_b - \boldsymbol{R}_{b\dot{b}} \, \boldsymbol{R}_{\dot{b}}^{-1} \, \boldsymbol{R}'_{b\dot{b}}\right] \boldsymbol{T}' \tag{5.4.26}$$

We have seen that knowing the MSV's of the elements of $\boldsymbol{e} = \boldsymbol{c} - \hat{\boldsymbol{c}}$ determines how to quantize and code them. For example, if the mean square error distortion criterion is used then the overall pel MSE is equal to the sum of the mean square quantization errors of elements of $\boldsymbol{e}$. If (as is almost always the case) the elements are approximately Laplacian, then the results of Section 5.3.2 can be applied straightforwardly. However, if more realistic distortion criteria

are used in order to exploit the properties of human vision, then other quantization strategies may have to be employed.

5.5 Other Block Coding

There are many other block coding methods besides transform coding and its variations. In this section we describe a few of them.

5.5.1 Vector Quantization and Variations

As we saw in Section 5.2.4, scalar quantization involves basically two operations, 1) *partitioning* the range of possible input values into a finite collection of subranges or subsets, and 2) for each subset choosing a *representative value* to be output when an input value is in that subrange, i.e., a member of that subset. With vector quantization,[5.5.1] the same two operations take place not in a one-dimensional scalar space, but in an N-dimensional vector space. Thus, if the vector space is partitioned into 2^{NR} subsets each having a corresponding representation value or *code vector,* then blocks $\boldsymbol{b}$ of N pels can be coded with RN bits per block or R bits per pel.

If we assume that filtering, gamma correction and other preliminary operations have already taken place, then it becomes plausible to define a difference distortion measure $d(\boldsymbol{b}-\tilde{\boldsymbol{b}})$ between an input vector $\boldsymbol{b}$ and a code vector $\tilde{\boldsymbol{b}}$. The problem of vector quantization is then to choose the partitioning and the code vectors in such a way as to minimize the overall distortion for the class of imagery to be handled.

Any of the difference distortion measures described previously can be used, at least in principle. These include mean square error, subjectively weighted error, frequency weighted error, etc. However, study of vector quantization of images has made headway only in recent years, and in practically all of these studies the mean square error (MSE) has been used exclusively, that is

$$d = \frac{1}{N}\,(\boldsymbol{b}-\tilde{\boldsymbol{b}})'\,(\boldsymbol{b}-\tilde{\boldsymbol{b}}) \tag{5.5.1}$$

$$= \frac{1}{N}\sum_{i=1}^{N}(b_i-\tilde{b}_i)^2$$

with the overall distortion being simply the expected value

$$D = E(d) \tag{5.5.2}$$

Utilization of non-difference distortions has not yet been attempted.

The design of image vector quantizers (VQ) generally requires a set of *training* pictures since accurate statistics are usually unavailable. The VQ code vectors and partitioning are then determined iteratively by repeated processing of the training set. Having designed the VQ, a block $\boldsymbol{b}$ of pels can be coded (for most distortion measures) by a simple nearest neighbor rule, that is, choose the code vector $\tilde{\boldsymbol{b}}$ that minimizes $d(\boldsymbol{b}-\tilde{\boldsymbol{b}})$ and transmit a code word index of NR bits to tell the receiver which code word to use. The decoder then simply displays the vector $\tilde{\boldsymbol{b}}$ as the coded representation.

The main advantage of VQ is the simple receiver structure, which consists only of a look up table or *codebook* containing the 2^{NR} code vectors. The disadvantages include coder complexity and the fact that images dissimilar to those in the training set may not be well represented by the code vectors in the codebook.

5.5.1a Design of VQ Codebook Using Training Images

One method proposed recently is the generalized Lloyd algorithm, also known as the LBG algorithm[5.5.2] or the *K-means* algorithm.[5.5.3] It requires an initial codebook and proceeds as follows.

1) Map the training blocks into the code words using the nearest neighbor rule. If the overall distortion of this mapping or quantization is low enough, quit. Otherwise,

2) For each code vector $\tilde{\boldsymbol{b}}$ determine the subset of training blocks that were mapped into it. Then replace $\tilde{\boldsymbol{b}}$ by another code word that reduces (ideally minimizes) the overall distortion for that subset of training blocks. Then go to step 1.

For the MSE distortion the best replacement code word in step 2 is simply the average of the corresponding subset of training blocks. For other distortions the best replacement code word may be much more difficult to compute. The algorithm terminates when the overall distortion stops decreasing significantly.

Note that for a given initial codebook the LBG algorithm only produces a *local* minimum distortion. Other initial codebooks may well converge to a better final codebook. In fact, choosing an initial codebook is a problem in itself. One plausible choice for an initial codebook is a subset of the training vectors. These should be chosen to be fairly dissimilar from each other, and obvious outliers should probably be avoided. Another initial codebook might be obtained simply by using a coarse scaler quantizer on the individual components of the training vectors and throwing out duplicates.

A third method for initial codebook selection called *splitting* starts with a single code word equal to the average of the training vectors. Another codeword is then generated by a small perturbation, and the LBG algorithm is invoked to optimize the two codewords (see Fig. 5.5.1). The process then continues in a similar manner, that is, given K codewords, generate $2K$ codewords by a small

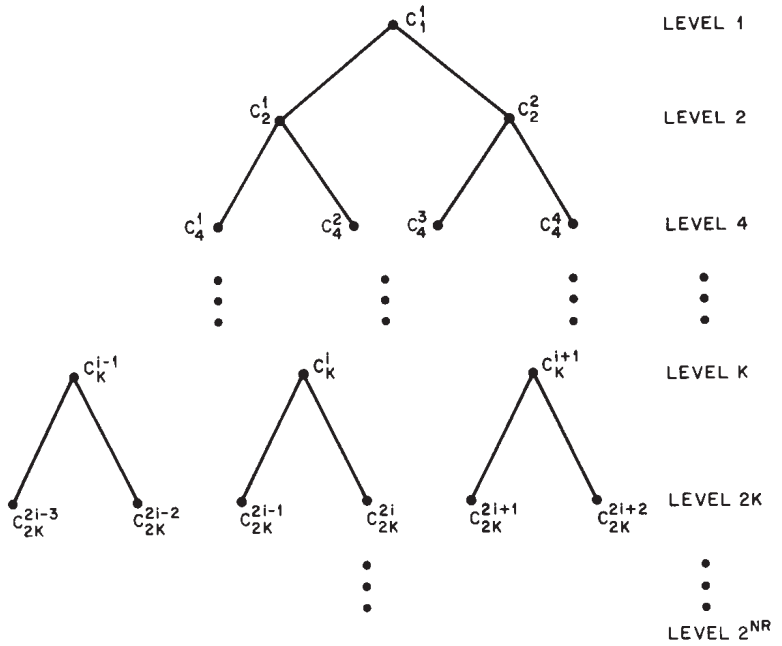


Fig. 5.5.1 The splitting method for generating a Vector Quantization codebook. At a given level K, each of the K code vectors is perturbed slightly to yield a set of $2K$ vectors, which is then optimized by the LBG algorithm to produce the codebook at level $2K$. The tree can also be used for coding. At each node C_K^i, the input vector is compared with code vectors C_{2K}^{2i-1} and C_{2K}^{2i}. The closer one is then chosen as the next node.

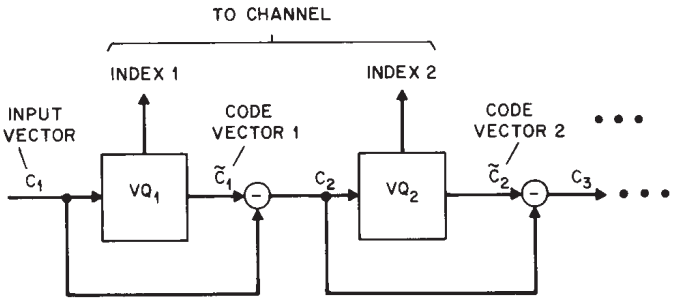


Fig. 5.5.2 Multistage Vector Quantization. At each state an error vector is computed, which is then used as the input to the next stage of VQ. The decoder merely computes a summation of the code vectors corresponding to the received indices.

perturbation of each, followed by the LBG algorithm to optimize them. Continue until 2^{NR} codewords are generated.

A problem with the LBG algorithm and its variants is its excessive convergence time in building the codebook. Techniques for reducing this time invariably tradeoff performance; however, since the LBG algorithm itself is only locally optimum this may not be much of a liability. One technique called *tree codebooks*[5.5.2] starts with one codeword and proceeds as in the splitting method above. However at each stage K of code book generation the set of training vectors is partitioned into subsets of training vectors that are mapped into the same code vector. Then at the next stage, as each code vector is perturbed slightly to produce another code vector, only those training vectors in the corresponding subset are used to optimize the resulting two vectors.

Another codebook generation technique developed by Equitz[5.5.3] is significantly faster with no apparent performance degradation. It is called the *Nearest Neighbor* or NN algorithm, and has computational complexity that grows only linearly with the size of the training set. The basic idea of this method is to start with the entire training set and, as a first step, merge the two vectors that are closest to each other into another vector equal to their mean. Successive application of this operation eventually reduces the set of vectors to the desired size, or alternatively raises the mean square quantization error to just short of an unacceptable level. The main computational efficiency of the NN algorithm is achieved by a preprocessing operation that partitions the training data into a *K-d Tree*. This enables not only much faster searching, but also multiple merges at each iteration.

The NN algorithm frequently produces a better codebook than the LBG algorithm, which shows the importance of initial codebook selection in the latter. On the other hand, an even better codebook is usually obtained by using the NN algorithm to derive an initial codebook for the LBG algorithm.

5.5.1b VQ Variants and Improvements

Several methods have been suggested for speeding up the coding operation, i.e., selecting the code vector that best approximates each input vector. Most of these include construction of a code tree of some type, which greatly speeds up the search, while sacrificing somewhat on performance. For example, the splitting technique tree codebook mentioned above for codebook generation can also be used for suboptimal coding. At each node a comparison is made between the input vector and the intermediate code vectors corresponding to that node in order to determine which branch to be traversed next (see Fig. 5.5.1). Similarly, the NN algorithm can be used to construct a code tree simply by letting the algorithm continue until only one code word is left. For a binary tree (two branches at each node) the number of comparisons is reduced from 2^{NR} to $2NR$, i.e., the computation time increases only linearly with the vector length. However, the storage requirement is essentially doubled compared with exhaustive search coding.

Multistage VQ (see Fig. 5.5.2) reduces both the search time and the storage requirement. At the first stage a low rate VQ is performed, and a code word index is transmitted. Then a first error vector is generated by subtracting the code vector from the original. This error vector is then coded by a different VQ, a code word index is transmitted and if necessary, a second error vector is generated. The receiver produces a decoded vector by simply summing up the code vectors corresponding to the received indices. In principle, coding error can be reduced in this way to any desired level. However, in practice the pels of the error vectors eventually become very uncorrelated and, thus, unsuited to efficient vector quantization.

Parameter extraction techniques, also called *Product Codes,* are capable of further reducing VQ coding complexity. For example, if the mean and variance of each input vector are computed and sent separately, then the mean can be subtracted and a gain factor applied to produce a zero mean, unity variance vector to be coded by VQ. The codebook for these vectors will presumably be much smaller than an equivalent quality simple VQ codebook for the original vectors. Also, the scalar parameters may be coded themselves, for example, by DPCM.

Block classification techniques may be used to good advantage in adaptive Vector Quantization. Here, blocks are classified into one of several classes according to spatial activity, and a different VQ codebook is used for each class. This approach not only produces codebooks that are better suited to the blocks they will be used to code, it also exhibits less sensitivity to changes in picture material. A drawback of this method is the overhead required to send the class of each block.

As with hybrid transform coding, prediction techniques can also be used with VQ. Simple predictive VQ would compute a prediction of the present block based on previously coded blocks just as in hybrid transform coding. Adjacent blocks either in the same frame or in the previous frame may be used in the prediction. The prediction error is then coded and sent by VQ in the same way as with nonpredictive VQ.

Vector quantization of color images raises the possibility of exploiting correlations between color components since, in principle, a block of $L \times L$ colored pels is simply an $L \times L \times 3$ array of numbers that can be rearranged as an $3L^2$ vector. VQ can then be carried out as before, except we have the ever present problem of finding a suitable distortion criterion. This approach can be quite efficient for block classifications techniques are used and each image to be coded contains relatively few colors. For then, the code books can be relatively small. Also, see Section 5.1.1a on color maps. However, if a wide variety of colors must be accommodated, then as we saw in Chapter 3 the *YIQ* color components are practically uncorrelated. In this case combined component coding is of little use, and separate component coding gives nearly optimum results.

5.5.1c Performance of VQ Codecs

The picture quality achievable by VQ coding depends on many factors, including the block size of the pel vectors, the amount of interpel correlation, the suitability of the codebook to the images to be coded, etc. VQ is useful mostly for low bit-rate coding of pictures, since high rates usually entail an impractical degree of complexity. For example, with blocks of size 4×4 pels, coding at a rate of one bit per pel implies a codebook containing 2^{16} code words. While this is not much of a problem for the decoder, the codebook search during coding can be prohibitively complex and lengthy, as can the process of constructing codebooks. Continuing the above example, with images of size 256×256 pels a training set of 16 pictures itself contains only 2^{16} blocks. This implies that a much larger training set is needed, thus further increasing the complexity of codebook generation.

Fig. 5.5.3 shows the results of VQ coding at 0.5 bits per pel using blocks of size 4×4 pels. The codebook was constructed using Equitz's NN algorithm and a training set of two images of size 512×512 pels. Note that the coding error of the training images is considerably smaller than for pictures outside the training set.

Optimizing the subjective image quality using vector quantization can only be described at present, as a worthy, but elusive goal. Mean square error is virtually the only criterion considered so far. However, it has been recognized that different types of picture material, e.g., low detail, texture, edges, should be coded and reproduced in different ways[5.5.4] and that vector quantization may be an ideal method for doing so.

5.5.2 Nonunitary Interpolative Transform Coding

Here we briefly discuss a somewhat different approach called nonunitary interpolative transform coding, which we illustrate by example. Suppose we partition the image into 4×4 blocks, with pels of each block labeled as follows:

$$\begin{matrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ M & N & O & P \end{matrix} \tag{5.5.3}$$

For each block, we then code and transmit the first pel A unchanged. We

Fig. 5.5.3 Pictures of size 512×512 pels are coded at 0.5 bits per pel using Vector Quantization with block size 4×4 pels [from H-M. Hang]. The training set consisted of the blocks of images a and b. Image c is outside the training set. a) *PSNR* = 31.0 dB. b) *PSNR* = 30.7 dB. c) *PSNR* = 26.8 dB.

then code and transmit pel P via DPCM using† pel $\tilde{A}$ as a prediction and then send pels D and M via DPCM using $(\tilde{A}+\tilde{P})/2$ as a prediction. The remaining pels will then be coded and transmitted via DPCM using interpolations of previously sent nearest neighbors as predictions. Thus, for example, pel B would be coded using as a prediction the interpolated value $(2\tilde{A}+\tilde{D})/3$, and pel C would then use $(\tilde{B}+\tilde{D})/2$ as a prediction. In this way, all predictions use only pels that have been previously quantized and transmitted, thus avoiding a possible accumulation of unwanted quantization error. If we arrange the block of pels in Eq. (5.5.3) in a column vector

$$\boldsymbol{b} = [A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P]' \tag{5.5.4}$$

then all of the aforementioned DPCM differential values can be computed by a 16×16 "transform" matrix. For example, the first six rows of the transform matrix

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
-\frac{1}{2} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} \\
-\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -\frac{1}{2} \\
-\frac{2}{3} & 1 & 0 & -\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & & & & & & & \vdots & & & & & & & & \\
 & & & & & & & & & & & & & & &
\end{bmatrix} \tag{5.5.5}$$

correspond to interpolative DPCM coding of pels A, P, D, M, B and C.

The approach described so far differs somewhat from simple transform coding in that each transform coefficient is now associated with a particular pel of the block $\boldsymbol{b}$. Moreover, after each coefficient in quantized, the associated pel of $\boldsymbol{b}$ is replaced by its quantized (tilde) value so that all predictions may be accurately computed at the receiver. Since the transform matrix is very sparse, containing mostly zeros, the complexity can be considerably less than with

† Tildes denote quantized values.

orthogonal transforms. In most cases, complexity increases only linearly with block size.

A generalization of this approach is possible if, in formulating the predictions, we do not worry about whether or not neighboring pels have already been transmitted. For example, with the transform matrix whose first five rows are

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ -\frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & & \cdots \\ 0 & -\frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & & \cdots \\ -\frac{1}{3} & 0 & 0 & 0 & 1 & -\frac{1}{3} & 0 & 0 & -\frac{1}{3} & & \cdots \end{bmatrix} \tag{5.5.6}$$

the pels A and D are coded as before. However, the predictions for pels B, C and E use, as yet, untransmitted pels. Thus, as in the case with simple transform coding, more than one coefficient is required in order to decode some pels.

In general, the transform matrix can be made completely previous-pel predictive, i.e., each transmitted coefficient enables the decoding of one pel of the block, if through some reordering of rows and/or columns† the transform matrix can be made lower-diagonal. In this case the inverse transform matrix is also lower-diagonal, which also implies that the transform is non-unitary. If the transform matrix cannot be diagonalized in this fashion, the non-unitary transform operation

$$\boldsymbol{c} = \boldsymbol{T}\boldsymbol{b} \tag{5.5.7}$$

will be able to make only partial use of quantized (tilde) pel values in computing the transform coefficients.

A unitary transformation can be constructed by defining the columns of $\boldsymbol{V}'$ to be the normalized eigenvectors of $\boldsymbol{T}$. In this case Eq. (5.5.7) can be written

$$\boldsymbol{V}\boldsymbol{c} = \boldsymbol{V}\boldsymbol{T}\boldsymbol{V}'\ \boldsymbol{V}\boldsymbol{b} \tag{5.5.8}$$

where $\boldsymbol{V}\boldsymbol{T}\boldsymbol{V}'$ is a diagonal matrix of eigenvalues. The resulting unitary transform is

† Reordering columns implies a similar reordering of pels in $\boldsymbol{b}$.

$$\boldsymbol{c}_u = \boldsymbol{V}\boldsymbol{T}^{-1}\boldsymbol{c} = \boldsymbol{V}\boldsymbol{b} \tag{5.5.9}$$

In most cases, little or no use of quantized (tilde) pel values will be possible with these unitary transforms. However, they still may be useful in certain adaptive coding situations where pel-to-pel correlations and dependencies change often, in a known way, and based upon this information it becomes possible to easily construct highly efficient interpolative transforms.

5.5.3 Block Truncation Coding (BTC)

The objective of this algorithm[5.5.5] is to quantize the pels of an $L \times L$ block in such a way as to preserve the mean and variance. In particular, a two-level quantizer is attractive because of its simplicity. The first step is to measure the mean and variance of the block

$$\bar{b} = \frac{1}{L^2} \sum_{i,j=1}^{L} b_{ij}$$

$$\sigma^2 = \frac{1}{L^2} \sum_{i,j=1}^{L} (b_{ij} - \bar{b})^2 \tag{5.5.10}$$

These are then quantized and transmitted. Next, the pels of the block are quantized to two levels using $\bar{b}$ as a threshold, i.e., a 1 or 0 is sent depending on whether b_{ij} is greater than or less than $\bar{b}$, respectively.

At the receiver the first step is to evaluate q, the number of 1's received for the block. The number of zeros is then $p = L^2 - q$. The next step is to evaluate the two quantizer levels that preserve mean and variance. These are easily shown to be

$$L_0 = \bar{b} - \sigma\sqrt{\frac{q}{p}}$$

$$L_1 = \bar{b} + \sigma\sqrt{\frac{p}{q}} \tag{5.5.11}$$

Finally, each pel is decoded to L_0 or L_1 depending on whether a 0 or 1 was received, respectively.

Using blocks of size 4×4 pels, and 8-bits each for $\bar{b}$ and σ, the coding bit-rate is 2-bits/pel. If $\bar{b}$ and σ are coded jointly using 10-bits, the data rate is 1.625 bits/pel.

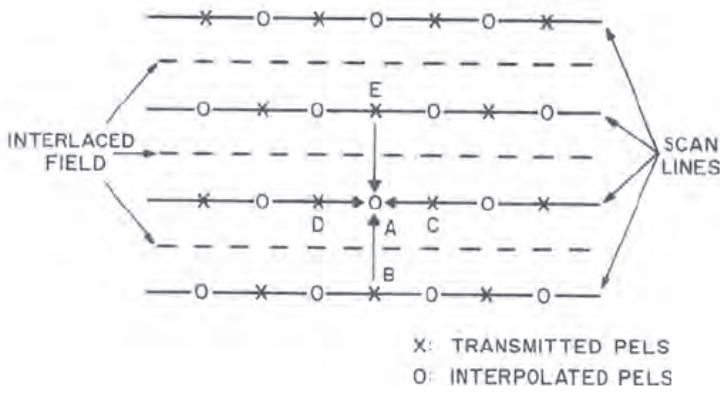


Fig. 5.6.1 An example of interpolative coding using 2:1 horizontal sub-sampling staggered from one line to the next.

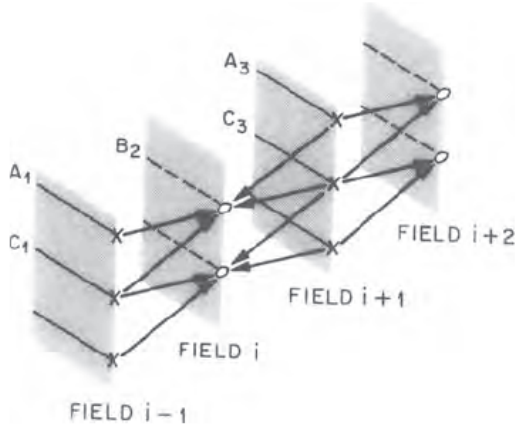


Fig. 5.6.2 Four-way averaging in which alternate fields (e.g. Field i, i+2) are not transmitted. At the receiver, the untransmitted fields are replaced by a four-way average of elements in the adjacent fields.

Decoded BTC images typically reproduce sharp edges of large objects fairly well. Also, random texture, although not reproduced with high accuracy, often looks good enough to be acceptable in many applications. However, in low detail areas of the picture where brightness and/or color vary slowly, there is often visible contouring and block-edge discontinuity due to the coarse quantization.

5.6 Miscellaneous Coding Methods

In this section we describe some other coding techniques that are currently under study. Some can be combined with methods already described, while others are stand-alone.

5.6.1 Interpolative and Extrapolative Coding

In interpolative and extrapolative coding, a subset of picture elements are first chosen for transmission. For this subset of pels any of the previously discussed coding schemes, such as PCM, DPCM or transform coding may be used. The pels that are not selected for transmission are reproduced by interpolation at the receiver using the information about the transmitted pels. In interpolative coding, some present and some future pels are transmitted and the rest are interpolated, whereas in extrapolative coding, pels in the immediate future are extrapolated from the past, one by one. The first pel for which the error of extrapolation goes above a previously set threshold causes the process of extrapolation to stop. That pel is then transmitted along with its address, and the following pels are extrapolated again as before.

Both interpolative and extrapolative methods can be divided into two classes: fixed and adaptive. In fixed interpolative methods, a fixed set of pels are selected for transmission and the rest are interpolated. Various types of samples can be chosen for transmission: For example, one may transmit every alternate sample, one sample out of four, every alternate scan line, every alternate field or frame, etc. The picture quality that results is a function of the number and type of samples that are dropped and the method of interpolation. Fig. 5.6.1 shows an example in which there is 2:1 subsampling along each scan line. The subsampling pattern is staggered from one line to the next and the dropped pels are interpolated by a four way average as shown by arrows, i.e., pel A is interpolated by averaging pels B, C, D, and E. More sophisticated interpolation can be used instead of linear weighted average. Interpolation based on higher degree polynomials or splines has been used, but experience shows that linear interpolation is quite effective and not much is gained by using polynomials of higher degree.

Adaptive interpolation can be more effective than fixed interpolation. As an example in Fig. 5.6.1, pel A may be reconstructed at the receiver by the following rule:

$$\hat{A} = \begin{cases} 0.5(C+D) , & \text{if } |C-D| \leq |B-E| \\ 0.5(B+E) , & \text{otherwise} \end{cases} \tag{5.6.1}$$

Such interpolations switch between horizontal, vertical, temporal or other directional averages depending upon the type of local correlation that exist in the data as measured by the transmitted pels.

An interpolative scheme that is commonly used in interframe coding is to drop alternate fields from transmission. This is shown in Fig. 5.6.2. Each pel in the dropped field is interpolated by taking a four way average of pels from adjacent fields that are spatially and temporally close. Such a four way average has the effect of reducing both the spatial as well as temporal resolution. It has been used effectively in interframe coding during buffer overloads in order to reduce the data rate.

Adaptive interpolative coding consists of three parts. a) choosing certain pels for transmission, b) constructing an interpolation (fixed or adaptive) of nontransmitted pels, and c) evaluating the interpolation error. If the error is below a certain preset threshold, then fewer pels are chosen for transmission; if it goes above the threshold, then a larger number of pels are chosen for transmission. The fewest number of pels that are needed to keep the interpolation error below the threshold is normally the desired characteristic. As an example, consider the video waveform along a scan line as shown in Fig. 5.6.3. Starting with sample A, one can go up to sample E and interpolate, using a straight line, all the in-between samples B, C, and D, keeping the interpolation error below the threshold. However, when samples A and F are chosen for transmission, then a simple straight line interpolation between pels A and F results in excessive interpolation error for pels B, C, D and E. Thus, pels A and E should be chosen for transmission, and pels B, C and D should be interpolated.

Several variations exist depending upon: a) the method used for transmission of pels such as A and E, b) interpolation and c) the method of judging the quality of interpolation. A more flexible variation is to treat this as a problem in waveform approximation in which values at certain pel positions (called knots) are transmitted and an approximation to the entire curve is made from the knot positions and the transmitted values (which may not coincide with the image intensities at the knot positions). The quality of interpolation is usually judged by the *MSE* between the interpolated signal and the true signal, since *MSE* is computationally straightforward. However, better results are obtained by using a quality criterion that is more closely related to human visual perception. As an example, better results are obtained by spatially filtering the error and then multiplying it by an appropriate masking function, before averaging the error over several pels. Unlike fixed interpolative coding, adaptive interpolative coding requires two types of information to be transmitted to the

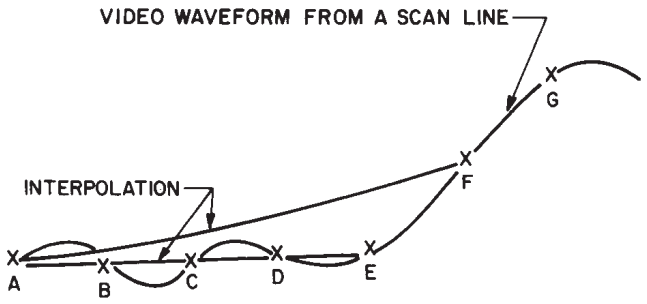


Fig. 5.6.3 An illustration of adaptive interpolation technique.

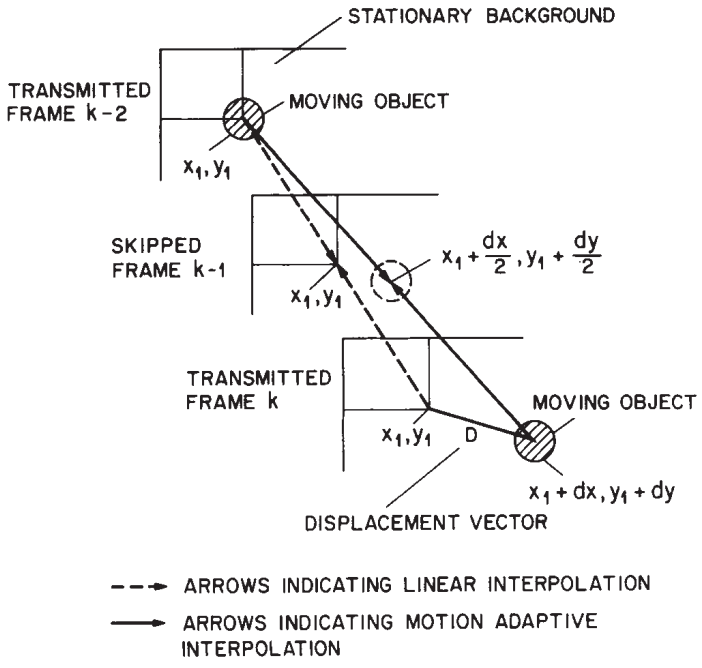


Fig. 5.6.4 Illustration of linear and motion interpolation. (from Musmann et al. [5.2.9])

receiver: a) addresses of pels chosen for transmission, and b) intensity of pels chosen for transmission.

The performance of the interpolative coders depends upon the technique used for transmitting samples and for interpolating. Interpolative coders show an improvement over nonadaptive DPCM and transform coding in terms of the number of bits required for a given picture quality. However, interpolative coders are inferior to both adaptive DPCM as well as adaptive transform techniques. Interpolative coders are useful for applications where a low bit rate is required, and where it might not be possible to encode each sample.

5.6.2 Motion Adaptive Interpolation

An adaptive interpolation that is very useful is called *motion adaptive* interpolation. Earlier in this section (Fig. 5.6.2) we described linear interpolation of dropped fields and remarked that four way linear average reduces spatial as well as temporal resolution. In some cases, it is possible to improve the quality of interpolation by using estimates of motion of objects in the scene. Fig. 5.6.4 shows an example where frame $(k-1)$ is skipped and is reconstructed at the receiver by using the frames transmitted at instants $(k-2)$ and (k). In this case, linear interpolation for a pel at location $(x_1 y_1)$ in frame $(k-1)$ would be a simple average of pels at the same location in frames $(k-2)$ and (k). This would blur the picture since, pels from background and moving object are averaged together. If however, the moving object was undergoing pure displacement as shown in the figure, then a pel at position $(x_1 + dx/2, y_1 + dy/2)$ in the skipped frame should be interpolated using pels (x_1, y_1) and (x_1+dx, y_1+dy) in frames $(k-2)$ and (k), respectively. This is indicated by arrows in the figure.

In order for such schemes to be successful, in addition to an accurate displacement estimator, it is necessary to segment the skipped frame into moving areas, uncovered area in the present frame relative to the previous frame and area to be uncovered in the next frame, so that proper interpolation can be applied. Obviously pels as well as coefficients used for interpolation of these different segments have to be different. This is a subject of current research. Preliminary simulations indicate that under moderate motion, a 3:1 frame dropping (i.e. skipping 2 out of 3 frames) and motion adaptive interpolation often gives reasonable motion rendition. In this as well as other interpolation schemes, since sometimes the interpolation may be inaccurate, techniques have been devised where the quality of interpolation is checked at the transmitter, and if the interpolation error is larger than a threshold, correction information is transmitted to the receiver. Due to unavoidable inaccuracies of the displacement estimator (e.g. complex translational and rotational motion) and the segmentation process, such side information is necessary to reduce artifacts that may otherwise be introduced due to faulty interpolation. Much more will be said about motion compensated interpolation in Chapter 9.

5.6.3 Pyramid Coding, Subband Coding

Another technique for image compression is called *pyramid* coding[5.6.1]. It represents the uncoded image as a series of band-pass images each sampled at successively lower rates. One implementation of this is to construct in the first stage a sequence of low-pass filtered images, $\{b_k(x_j,y_i)\}$, $k = 1 \ldots n$, such that (ignoring edge effects)

$$b_{k+1}(x_j, y_i) = \sum_{m=-p}^{+p} \sum_{n=-p}^{+p} h(m,n)\, b_k(x_{2j+n}, y_{2i+m}) \qquad (5.6.2)$$

where weights $\{h(m,n)\}$ are chosen as an approximate Gaussian-looking function, and $b_0(x_j,y_i)$ is the original uncoded image. For simplicity, $h(m,n)$ is made separable

$$h(m,n) = h(m)\, h(n) \qquad (5.6.3)$$

For $p=2$, the weights are taken to be $h(0) = 0.4$, $h(1) = h(-1) = 0.25$ and $h(2) = h(-2) = 0.05$. In this case, $h(m,n)$ is a cartesian product of two identical symmetric triangular functions.* Image $b_{k+1}(\cdot)$ is a low-pass version of $b_k(\cdot)$ that has been subsampled at a lower rate (in this case by a factor of two).

The next stage of coding is to generate a sequence of error images by first expanding the low-pass, subsampled image $b_{k+1}(\cdot)$ by interpolation and then subtracting it from $b_k(\cdot)$. Thus,

$$L_k\,(x_j,y_i) = b_k(x_j,y_i) - b_{k+1,e}(x_j,y_i) \qquad (5.6.4)$$

where the interpolated pels of the expanded version $b_{k+1,e}(\cdot)$ are given by

$$b_{k+1,e}(x_j,y_i) = 4 \sum_{m=-2}^{+2} \sum_{n=-2}^{+2} h(m,n)\, b_{k+1}\left[x\,\frac{j-n}{2},\ y\,\frac{i-m}{2}\right] \qquad (5.6.5)$$

In the above equation, only integer subscripts of x and y are to be included in the summations.

Thus, we have created a sequence of band-pass filtered images, $L_k(\cdot)$. If we imagine these images as stacked one above the other, the result is a tapering pyramid type of data structure. The compression is then accomplished by quantizing $b_n(\cdot)$ followed by the $L_k(\cdot)$'s and then entropy coding the quantized outputs. Image $b_n(\cdot)$ may, in fact, contain only one pel. At the receiver, inverse operations can be performed easily. Simulations show that performance of about 1 bit/pel can be achieved for reasonable picture quality. In addition to the good compression ratio, the scheme also lends itself nicely to progressive

---

* See Fig. 1.3.6a.

transmission. In that case, the topmost level of the pyramid is sent first, and expanded by interpolation in the receiver to form an initial coarse image. This is followed by the transmission of the next lower level of the pyramid, which is added after expansion to the existing image in the receiver, thereby increasing the quality of the image at the receiver. The principal advantage is that the computations are simple, local, and may be performed in parallel. Moreover, the same computations are iterated to build the sequence of images constituting the pyramid.

In the procedure described above, the $L_k(\cdot)$ may be obtained not interactively, but instead as the output of n parallel bandpass filters. In this case the compression algorithm is called *subband coding*.[5.6.2] As above, the $L_k(\cdot)$ are quantized and coded separately, perhaps using run-length coding for strings of zero values. The decoded image is obtained by simply adding the received bandpass images.

5.7 Coding of Graphics

We have so far looked at coding of multilevel (gray level and color) pictures. While many of the methods of multilevel coding are applicable to two-level (black = 0 and white = 1) picture signals, other methods that are unique to two level signals tend to be more efficient. In this section we discuss such techniques. Coding methods of this section may be divided into two broad classes: (a) those that allow exact reproduction of the picture at the receiver assuming that there were no transmission errors, (sometimes called *information-lossless* or exact coding techniques) and (b) methods that make an approximation of the picture (referred to as *information-lossy* or approximate coding).

5.7.1 Exact Coding

We consider four methods of exact coding. These are: (a) run-length coding, (b) predictive coding, (c) line-to-line predictive differential coding, and (d) block coding. This classification is somewhat artificial since there are several coding schemes that combine the above methods into one. Run-length coding is primarily one-dimensional, whereas the other schemes are primarily two-dimensional.

5.7.1a Run-Length Coding

In run-length coding, a run of consecutive pels of the same color (i.e. black or white) is combined together and represented by a single code word for transmission. For example, in Fig. 5.7.1, the length of each run (black or white pels) is represented by binary words. Since runs of black and white pels alternate, except for the first run of each line, the color of the run need not be transmitted. Statistics of such run-lengths vary from document-to-document. In general, they are highly nonuniform (see Section 3.5). Variable length coding of

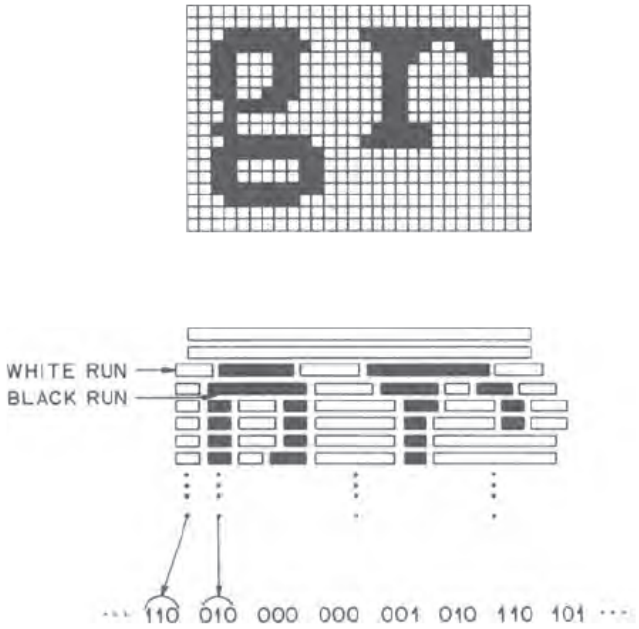


Fig. 5.7.1 A digitized black and white image on the pel grid and its representation in terms of horizontal runs.

DOC	$\bar{r}_w$	$\bar{r}_b$	H_w	H_b	Q_{max}
1	156.3	6.793	5.451	3.592	18.03
2	257.1	14.31	8.163	4.513	21.41
3	89.81	8.515	5.688	3.572	10.62
4	39.00	5.674	4.698	3.124	5.712
5	79.16	6.986	5.740	3.328	9.500
6	138.5	8.038	6.204	3.641	14.89
7	45.32	4.442	5.894	3.068	5.553
8	85.68	70.87	6.862	5.761	12.40

Fig. 5.7.2 Run-length distribution for the CCITT documents of Fig. 2.6.6. $\bar{r}_w$ and $\bar{r}_b$ are the average lengths of the white and black runs, respectively, and H_w and H_b are the corresponding entropies in bits/run. Q_{max} is the maximum compression ratio, i.e., it is the inverse of entropy, where entropy is expressed in bits/pel. (from Hunter and Robinson [5.7.1])

run-lengths takes advantage of the nonuniform probability distribution of run-lengths. Moreover, since the statistics of 0's are usually quite different from those of 1's, different code tables are often used, one for runs of 0's and another for 1's. Most of the variable length codes have been based on Huffman's procedure.

Since most documents have more than a thousand pels per line (the CCITT standard documents have 1728 as shown in Chapter 2), code tables could, in principle, contain a large number of code words. However, to avoid this complexity, a *modified* Huffman code is generally used in practical coders. In this modified Huffman code, every run-length greater than a given value (say l) is broken into two run-lengths: an initial run-length having a value Nl (where N is an integer) and a terminating run-length having a value between 0 and $(l-1)$. This reduces the number of code words in the table and the decoder implementation. Such a remapping of run-lengths modifies the run-length code statistics and results in a slight reduction of compression efficiency. Another advantage of the modified Huffman code is its ability to handle run-lengths of arbitrary size by proper extensions. In Chapter 6, we will give an example of the modified Huffman code that has been chosen as the CCITT standard.

Fig. 5.7.2 gives some relevant statistics that can be used to judge the efficiency of one-dimensional run-length codes for the CCITT standard images that are shown as Fig. 2.6.6. It is clear that for documents that have black information on a white background, the average white run-length ($\bar{r}_w$) is much larger than the average black run-length ($\bar{r}_b$). The compression ratio (entropy)$^{-1}$ also varies widely depending upon the document. On the average the compression ratio is about 12.

5.7.1b Predictive Coding

This is quite similar to predictive coding for gray level pictures. There are, however, important differences in the way a predictor is constructed and how the prediction errors are encoded for transmission, e.g., linear predictive coding is not used. The first step in the prediction process is to choose several previously transmitted surrounding picture elements to be used for prediction of the present element. As an example, consider the seven previous pels A, B, C, D, E, F, G for predicting the present pel X as shown in Fig. 5.7.3. Since each of these pels can take on two values, they together define a possible set of 128($= 2^7$) patterns called *states*. The maximum likelihood prediction $\hat{X}(S_i)$ for a given state S_i is given by

$$\hat{X}(S_i) = \begin{cases} \text{black , if} \quad P(X = \text{black} \mid S = S_i) > 0.5 \\ \\ \text{white, otherwise} \end{cases} \tag{5.7.1}$$

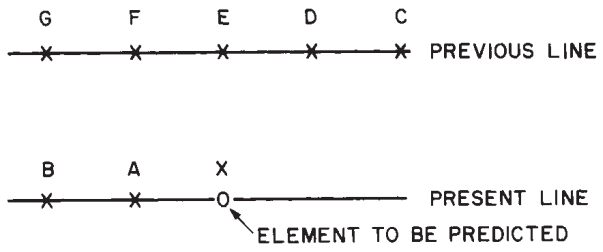


Fig. 5.7.3 The configuration of the seven elements constituting the state that is used to construct the predictor given in Fig. 5.7.4.

		FORWARD DIRECTION				REVERSE DIRECTION		
No.	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)
0	00000 00X	0	0.999	G	00000 X00	0	0.999	G
1	00000 01X	1	0.834	B	00000 X10	1	0.833	B
2	00000 10X	0	0.982	G	00000 X01	0	0.983	G
3	00000 11X	1	0.767	B	00000 X11	1	0.770	B
4	00001 00X	0	0.965	G	10000 X00	0	0.963	G
5	00001 01X	1	0.931	G	10000 X10	1	0.917	G
6	00001 10X	0	0.918	G	10000 X01	0	0.928	G
7	00001 11X	1	0.924	G	10000 X11	1	0.900	G
8	00010 00X	0	0.776	B	01000 X00	0	0.745	B
9	00010 01X	1	0.951	G	01000 X10	1	0.939	G
10	00010 10X	0	0.673	B	01000 X01	0	0.724	B
11	00010 11X	1	0.960	G	01000 X11	1	0.959	G
12	00011 00X	0	0.833	B	11000 X00	0	0.846	B
13	00011 01X	1	0.985	G	11000 X10	1	0.983	G
14	00011 10X	0	0.787	B	11000 X01	0	0.763	B
15	00011 11X	1	0.973	G	11000 X11	1	0.955	G
16	00100 00X	0	0.602	B	00100 X00	0	0.607	B
17	00100 01X	1	0.948	G	00100 X10	1	0.979	G
18	00100 10X	0	0.617	B	00100 X01	0	0.540	B
19	00100 11X	1	0.936	G	00100 X11	1	0.945	G
20	00101 00X	0	0.629	B	10100 X00	0	0.640	B
21	00101 01X	1	0.793	B	10100 X10	1	0.878	B
22	00101 10X	1	0.571	B	10100 X01	1	0.545	B
23	00101 11X	1	0.972	G	10100 X11	1	0.886	B
24	00110 00X	1	0.819	B	01100 X00	1	0.784	B
25	00110 01X	1	0.992	G	01100 X10	1	0.994	G
26	00110 10X	1	0.623	B	01100 X01	1	0.656	B
27	00110 11X	1	0.967	G	01100 X11	1	0.976	G
28	00111 00X	1	0.649	B	11100 X00	1	0.639	B

Fig. 5.7.4a State-dependent prediction and its quality for 128 states based on seven surrounding pels. Forward and reverse directions correspond to scanning from left to right and vice versa. States 0–28.

		FORWARD DIRECTION				REVERSE DIRECTION		
No.	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)
87	10101 11X	1	0.846	B	10101 X11	1	0.860	B
88	10110 00X	0	0.551	B	01101 X00	0	0.510	B
89	10110 01X	1	0.912	G	01101 X10	1	0.889	B
90	10110 10X	1	0.637	B	01101 X01	1	0.505	B
91	10110 11X	1	0.921	G	01101 X11	1	0.881	B
92	10111 00X	0	0.712	B	11101 X00	0	0.767	B
93	10111 01X	1	0.959	G	11101 X10	1	0.900	G
94	10111 10X	1	0.561	B	11101 X01	1	0.528	B
95	10111 11X	1	0.959	G	11101 X11	1	0.933	G
96	11000 00X	0	0.991	G	00011 X00	0	0.983	G
97	11000 01X	1	0.795	B	00011 X10	1	0.812	B
98	11000 10X	0	0.997	G	00011 X01	0	0.998	G
99	11000 11X	0	0.711	B	00011 X11	0	0.709	B
100	11001 00X	0	0.978	G	10011 X00	0	0.965	G
101	11001 01X	1	0.795	B	10011 X10	1	0.826	B
102	11001 10X	0	0.982	G	10011 X01	0	0.989	G
103	11001 11X	0	0.576	B	10011 X11	0	0.600	B
104	11010 00X	0	0.894	B	01011 X00	0	0.892	B
105	11010 01X	1	0.686	B	01011 X10	1	0.625	B
106	11010 10X	0	0.673	B	01011 X01	0	0.808	B
107	11010 11X	1	0.703	B	01011 X11	1	0.664	B
108	11011 00X	0	0.942	G	11011 X00	0	0.958	G
109	11011 01X	1	0.810	B	11011 X10	1	0.735	B
110	11011 10X	0	0.914	G	11011 X01	0	0.951	G
111	11011 11X	1	0.617	B	11011 X11	1	0.602	B
112	11100 00X	0	0.934	G	00111 X00	0	0.885	B
113	11100 01X	1	0.922	G	00111 X10	1	0.941	G
114	11100 10X	0	0.975	G	00111 X01	0	0.909	G
115	11100 11X	1	0.806	B	00111 X11	1	0.805	B

Fig. 5.7.4b States 29–57.

		FORWARD DIRECTION				REVERSE DIRECTION		
No.	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)
58	01110 10X	0	0.594	B	01110 X01	1	0.565	B
59	01110 11X	1	0.950	G	01110 X11	1	0.976	G
60	01111 00X	1	0.530	B	11110 X00	1	0.502	B
61	01111 01X	1	0.996	G	11110 X10	1	0.996	G
62	01111 10X	0	0.768	B	11110 X01	0	0.615	B
63	01111 11X	1	0.961	G	11110 X11	1	0.975	G
64	10000 00X	0	0.996	G	00001 X00	0	0.996	G
65	10000 01X	1	0.768	B	00001 X10	1	0.637	B
66	10000 10X	0	0.997	G	00001 X01	0	0.999	G
67	10000 11X	0	0.648	B	00001 X11	0	0.665	B
68	10001 00X	0	0.982	G	10001 X00	0	0.986	G
69	10001 01X	1	0.757	B	10001 X10	1	0.689	B
70	10001 10X	0	0.988	G	10001 X01	0	0.991	G
71	10001 11X	1	0.614	B	10001 X11	1	0.607	B
72	10010 00X	0	0.902	G	01001 X00	0	0.890	B
73	10010 01X	1	0.625	B	01001 X10	1	0.682	B
74	10010 10X	0	0.717	B	01001 X01	0	0.843	B
75	10010 11X	1	0.880	B	01001 X11	1	0.828	B
76	10011 00X	0	0.924	G	11001 X00	0	0.937	G
77	10011 01X	1	0.748	B	11001 X10	1	0.767	B
78	10011 10X	0	0.906	G	11001 X01	0	0.927	G
79	10011 11X	1	0.826	B	11001 X11	1	0.789	B
80	10100 00X	0	0.879	B	00101 X00	0	0.875	B
81	10100 01X	1	0.660	B	00101 X10	1	0.719	B
82	10100 10X	0	0.531	B	00101 X01	1	0.595	B
83	10100 11X	1	0.882	B	00101 X11	1	0.930	G
84	10101 00X	0	0.897	B	10101 X00	0	0.768	B
85	10101 01X	1	0.739	B	10101 X10	1	0.700	B
86	10101 10X	1	0.623	B	10101 X01	1	0.611	B

Fig. 5.7.4c States 58–86.

		FORWARD DIRECTION				REVERSE DIRECTION		
No.	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)
29	00111 01X	1	0.996	G	11100 X10	1	0.998	G
30	00111 10X	1	0.524	B	11100 X01	1	0.529	B
31	00111 11X	1	0.985	G	11100 X11	1	0.987	G
32	01000 00X	0	0.971	G	00010 X00	0	0.949	G
33	01000 01X	1	0.522	B	00010 X10	0	0.507	B
34	01000 10X	0	0.906	G	00010 X01	1	0.860	B
35	01000 11X	1	0.636	B	00010 X11	1	0.646	B
36	01001 00X	0	0.934	G	10010 X00	1	0.899	B
37	01001 01X	1	0.503	B	10010 X10	1	0.753	B
38	01001 10X	0	0.556	B	10010 X01	0	0.800	B
39	01001 11X	1	0.751	B	10010 X11	1	0.692	B
40	01010 00X	0	0.854	B	01010 X00	0	0.885	B
41	01010 01X	1	0.583	B	01010 X10	1	0.635	B
42	01010 10X	0	0.684	B	01010 X01	1	0.529	B
43	01010 11X	1	0.813	B	01010 X11	1	0.878	B
44	01011 00X	0	0.910	G	11010 X00	0	0.864	B
45	01011 01X	0	0.516	B	11010 X10	1	0.730	B
46	01011 10X	0	0.645	B	11010 X01	1	0.526	B
47	01011 11X	1	0.837	B	11010 X11	1	0.796	B
48	01100 00X	0	0.661	B	00110 X00	0	0.533	B
49	01100 01X	1	0.973	G	00110 X10	1	0.983	G
50	01100 10X	0	0.895	B	00110 X01	0	0.692	B
51	01100 11X	1	0.759	B	00110 X11	1	0.838	B
52	01101 00X	1	0.783	B	10110 X00	0	0.697	B
53	01101 01X	1	0.868	B	10110 X10	1	0.941	G
54	01101 10X	1	0.765	B	10110 X01	0	0.596	B
55	01101 11X	1	0.729	B	10110 X11	1	0.827	B
56	01110 00X	1	0.726	B	01110 X00	1	0.739	B
57	01110 01X	1	0.997	G	01110 X10	1	0.996	G

Fig. 5.7.4d States 87–115.

		FORWARD DIRECTION				REVERSE DIRECTION		
No.	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)	State Configuration	Prediction	Probability of Correct Prediction	Goodness (G or B)
116	11101 00X	0	0.951	G	10111 X00	0	0.900	G
117	11101 01X	1	0.885	B	10111 X10	1	0.910	G
118	11101 10X	0	0.920	G	10111 X01	0	0.908	G
119	11101 11X	1	0.748	B	10111 X11	1	0.753	B
120	11110 00X	0	0.756	B	01111 X00	0	0.724	B
121	11110 01X	1	0.975	G	01111 X10	1	0.972	G
122	11110 10X	0	0.849	B	01111 X01	0	0.813	B
123	11110 11X	1	0.929	G	01111 X11	1	0.926	G
124	11111 00X	0	0.830	B	11111 X00	0	0.835	B
125	11111 01X	0	0.957	G	11111 X10	0	0.953	G
126	11111 10X	0	0.848	B	11111 X01	0	0.847	B
127	11111 11X	1	0.989	G	11111 X11	1	0.989	G

Fig. 5.7.4e States 116–127.

PEL ROW	ℓ	$\ell+1$	$\ell+2$	$\ell+3$	
	8	136	40	168	LINE m
	200	72	232	104	LINE m + 1
	56	184	24	152	LINE m + 2
	248	120	216	88	LINE m + 3

Fig. 5.7.5 A 4×4 dither matrix for an image with intensity in the range 0–255 (8 bits). See Chapter 1.

where $P(\cdot|\cdot)$ is the experimentally determined conditional probability. In general, $P(\cdot|\cdot)$ is determined by computing histograms over a given set of documents. Using the 8 CCITT standard documents, the relevant statistics and a maximum likelihood predictor are given for each of the 128 states in Fig. 5.7.4. One obvious property of such a prediction scheme is that the probability of correct prediction is always higher than 0.5.

In the case of two-level pictures that are dithered to give the appearance of gray-level as described in Chapter 2, X can be predicted by including the value of the dither matrix in the definition of state. As an example, if we use a 4×4 dither matrix as shown in Fig. 5.7.5 there are 16 values of the dither matrix $\{D_{ij}\}$, $i, j = 1, \ldots, 4$. Using four surrounding elements A, D, E, and F as in Fig. 5.7.3 as well as the dither matrix value D_X, the state S can be defined by the five-tuple $S = (D_X, A, D, E, F)$. There are 256 possible states $\{S_i\}$. For a given pel, the maximum likelihood prediction is developed again as the most probable one given that a particular state has occurred. An example of such a predictor is given in Fig. 5.7.6, which shows only the first 54 states. The predictor, optimized for a particular picture, does show variation from picture-to-picture as a result of the nonstationarity of the image statistics.

For either two-level or dithered images, the next step in predictive coding is to compute the prediction error, which is obtained by a simple exclusive-or of the input sample and the prediction. The prediction error is then run-length coded, generally in the same direction as the scanning. Here again, variable length coding with two sets of code tables (one for 0's, and the other for 1's) is used.

Statistics of run-lengths of prediction errors conditioned on a particular state vary significantly from state to state. It is therefore more desirable to encode the conditional run-lengths, using a code table that is tailored to the statistics of that state. This is akin to state space coding as described in Chapter 3. Fig. 5.7.7 shows the conditional run-lengths for three states S_0, S_1 and S_2 for a scan line of hypothetical data. Here, instead of run-lengths of prediction error along the scan line, prediction errors of consecutive pels belonging to a particular state (or a group of states) define run-lengths. Complexity can be reduced to some extent by using only two groups of states and assigning a code table for each of these groups as explained below.

Another method of using the variation of conditional run-length statistics is *reordering*. In the reordering technique (see Chapter 3) the pels in a line are rearranged according to their state so as to improve the performance of the run-length coder. This may be done with only one code table or many code tables as in conditional run-length coding of the previous paragraph. Fig. 5.7.4 shows that the probability of making a correct prediction varies a great deal from state-to-state. For example, state 0 has a probability of correct prediction of 0.999, whereas, for state 45 it is 0.57. Thus, some states can be classified as *good* with respect to their ability to predict the next pel, and the others can be classified as

	STATE DEFINITION					PICTURE: KAREN			PICTURE: ENGINEERING DRAWING			PICTURE: HOUSE		
STATE NUMBER	Dither Matrix	F	E	D	A	Total Elements in State	Prediction	Probability of Prediction Error	Total Elements in State	Prediction	Probability of Prediction Error	Total Elements in State	Prediction	Probability of Prediction Error
1	8	0	0	0	0	4026	1	0.018	3197	1	0.041	6225	1	0.000
2	8	0	0	0	1	12	1	0.000	197	1	0.000	0	0	0.000
3	8	0	0	1	0	77	1	0.000	760	1	0.011	32	1	0.000
4	8	0	0	1	1	2	1	0.000	97	1	0.000	0	0	0.000
5	8	0	1	0	0	0	0	0.000	0	0	0.000	0	0	0.000
6	8	0	1	0	1	0	0	0.000	0	0	0.000	0	0	0.000
7	8	0	1	1	0	0	0	0.000	0	0	0.000	0	0	0.000
8	8	0	1	1	1	0	0	0.000	0	0	0.000	0	0	0.000
9	8	1	0	0	0	2590	1	0.000	1489	1	0.003	5665	1	0.000
10	8	1	0	0	1	43	1	0.000	429	1	0.000	0	0	0.000
11	8	1	0	1	0	6121	1	0.000	2342	1	0.002	4204	1	0.000
12	8	1	0	1	1	3258	1	0.000	7618	1	0.000	3	1	0.000
13	8	1	1	0	0	0	0	0.000	0	0	0.000	0	0	0.000
14	8	1	1	0	1	0	0	0.000	0	0	0.000	0	0	0.000
15	8	1	1	1	0	0	0	0.000	0	0	0.000	0	0	0.000
16	8	1	1	1	1	0	0	0.000	0	0	0.000	0	0	0.000
17	24	0	0	0	0	2938	1	0.279	3259	1	0.417	4217	1	0.047
18	24	0	0	0	1	4	1	0.000	108	1	0.000	0	0	0.000
19	24	0	0	1	0	61	1	0.000	785	1	0.135	10	1	0.000
20	24	0	0	1	1	0	0	0.000	32	1	0.000	0	0	0.000
21	24	0	1	0	0	0	0	0.000	2	0	0.000	0	0	0.000
22	24	0	1	0	1	0	0	0.000	0	0	0.000	0	0	0.000
23	24	0	1	1	0	0	0	0.000	0	0	0.000	0	0	0.000
24	24	0	1	1	1	1	1	0.000	0	0	0.000	0	0	0.000
25	24	1	0	0	0	2302	1	0.003	1335	1	0.061	5193	1	0.000
26	24	1	0	0	1	40	1	0.025	267	1	0.011	0	0	0.000
27	24	1	0	1	0	0185	1	0.006	5154	1	0.009	6984	1	0.000

Fig. 5.7.6 Definition of state based on four surrounding pels and the value of the dither matrix. State-dependent prediction and its quality is given for three dithered pictures. Only the first 54 states are shown. (a) States 1–27.

STATE NUMBER	STATE DEFINITION					PICTURE: KAREN			PICTURE: ENGINEERING DRAWING			PICTURE: HOUSE		
	Dither Matrix	F	E	D	A	Total Elements in State	Prediction	Probability of Prediction Error	Total Elements in State	Prediction	Probability of Prediction Error	Total Elements in State	Prediction	Probability of Prediction Error
28	24	1	0	1	1	1181	1	0.007	5442	1	0.001	0	0	0.000
29	24	1	1	0	0	0	0	0.000	0	0	0.000	0	0	0.000
30	24	1	1	0	1	0	0	0.000	0	0	0.000	0	0	0.000
31	24	1	1	1	0	3	1	0.000	0	0	0.000	0	0	0.000
32	24	1	1	1	1	39	1	0.000	0	0	0.000	0	0	0.000
33	40	0	0	0	0	4038	1	0.403	3096	0	0.193	6317	1	0.113
34	40	0	0	0	1	24	1	0.000	396	1	0.063	9	1	0.000
35	40	0	0	1	0	2502	1	0.002	1327	1	0.251	5641	1	0.000
36	40	0	0	1	1	70	1	0.000	587	1	0.027	24	1	0.000
37	40	0	1	0	0	0	0	0.000	0	0	0.000	0	0	0.000
38	40	0	1	0	1	0	0	0.000	1	1	0.000	0	0	0.000
39	40	0	1	1	0	0	0	0.000	0	0	0.000	0	0	0.000
40	40	0	1	1	1	2	1	0.000	0	0	0.000	0	0	0.000
41	40	1	0	0	0	33	1	0.091	333	1	0.339	42	1	0.000
42	40	1	0	0	1	47	1	0.021	513	1	0.226	16	1	0.000
43	40	1	0	1	0	1996	1	0.001	1187	1	0.127	2319	1	0.000
44	40	1	0	1	1	7090	1	0.000	6774	1	0.002	1888	1	0.000
45	40	1	1	0	0	0	0	0.000	0	0	0.000	0	0	0.000
46	40	1	1	0	1	0	0	0.000	0	0	0.000	0	0	0.000
47	40	1	1	1	0	15	1	0.000	1	1	0.000	0	0	0.000
48	40	1	1	1	1	379	1	0.000	41	1	0.000	0	0	0.000
49	56	0	0	0	0	2943	0	0.271	3024	0	0.082	4136	0	0.487
50	56	0	0	0	1	9	1	0.000	229	1	0.066	0	0	0.000
51	56	0	0	1	0	2274	1	0.047	1349	1	0.375	5146	1	0.000
52	56	0	0	1	1	6	1	0.000	265	1	0.049	0	0	0.000
53	56	0	1	0	0	0	0	0.000	0	0	0.000	0	0	0.000
54	56	0	1	0	1	0	0	0.000	0	0	0.000	0	0	0.000

Fig. 5.7.6b States 28–54.

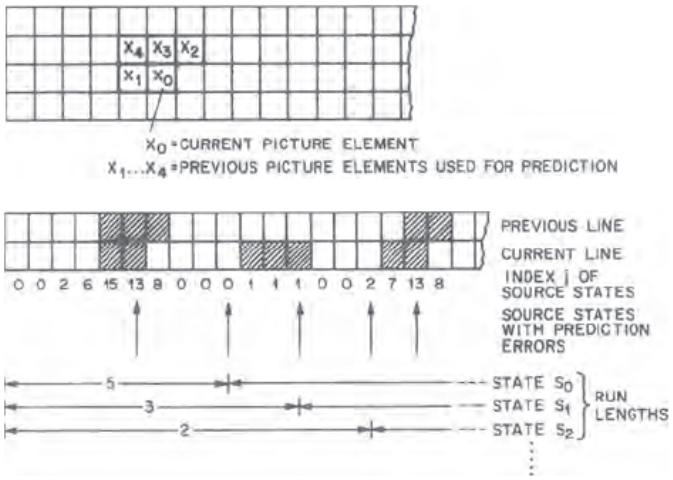


Fig. 5.7.7 Example of conditional run-length coding for three states S_0, S_1 and S_2. A conditional run is defined as a run of consecutive pels belonging to the same state. Conditional runs of length 5, 3 and 2 are shown for state S_0, S_1 and S_2.

bad. If the prediction errors for all of the good states are grouped together, then the run-lengths corresponding to prediction errors of these good states will be relatively long, and consequently a higher compression ratio can be achieved. To illustrate such a good/bad reordering process, consider a memory of 1728 cells, numbered from 1 to 1728 (equal to the number of pels per line). We classify good states as those for which the probability of correct prediction conditioned on that state is greater than or equal to a given threshold (called goodness threshold); all the other states are bad. In the process of reordering, if the first pel of the present line has a state that is classified as good, we put the prediction error corresponding to it in memory cell 1. If, on the other hand, the state is classified as bad, we put the prediction error corresponding to it in memory cell 1728. We continue in this manner: the prediction for the ith element of the present line is put in the unfilled memory cell with the smallest or the largest index, depending upon whether the state corresponding to the ith element is good or bad, respectively. When the memory is filled, the cells are read in numerical order, and the contents are run-length coded as usual. At the receiver the ordered data can be unscrambled uniquely (in the absence of transmission error), since the ordering sequence can be derived by the receiver.

Fig. 5.7.4 shows the classification of states into good and bad using a goodness threshold of 0.9. Fig. 5.7.8 shows one of the CCITT images, the corresponding prediction errors, and the good/bad reordered prediction errors, (a pel with prediction error is reproduced as black, and a pel without prediction error is reproduced as white). It is seen that the average run-length of the reordered data is much higher than that of the unordered data, resulting in better compression. Schemes have been developed in which only one code table is used for coding the run-length of the ordered data. Higher coding efficiency may be obtained by using separate code tables for runs in good and bad regions. The ordering process for dithered two-level images is identical to the one described above, with the proper definition of state, which includes the value of the dither matrix.

The performance of various predictive coding schemes is given in Fig. 5.7.9 for the eight CCITT standard images, where only entropies in bits/pel are given. The predictor used for these calculations consisted of four pels A, D, E, and F of Fig. 5.7.3 and the prediction rule was optimized for each picture. One dimensional run-length coding has a compression ratio between 5.7 and 22.4, whereas good/bad reordering the prediction errors, increases this ratio to 7.1 and 37.5. Using separate code tables for the good and bad region, the compression ratio can be increased by about 4–8 percent. Improvements of similar nature are obtained for dithered pictures.

5.7.1c Line-to-Line Run-Length Difference Coding

This scheme takes into account correlation between run-lengths in successive lines. Instead of transmitting run-lengths, differences between

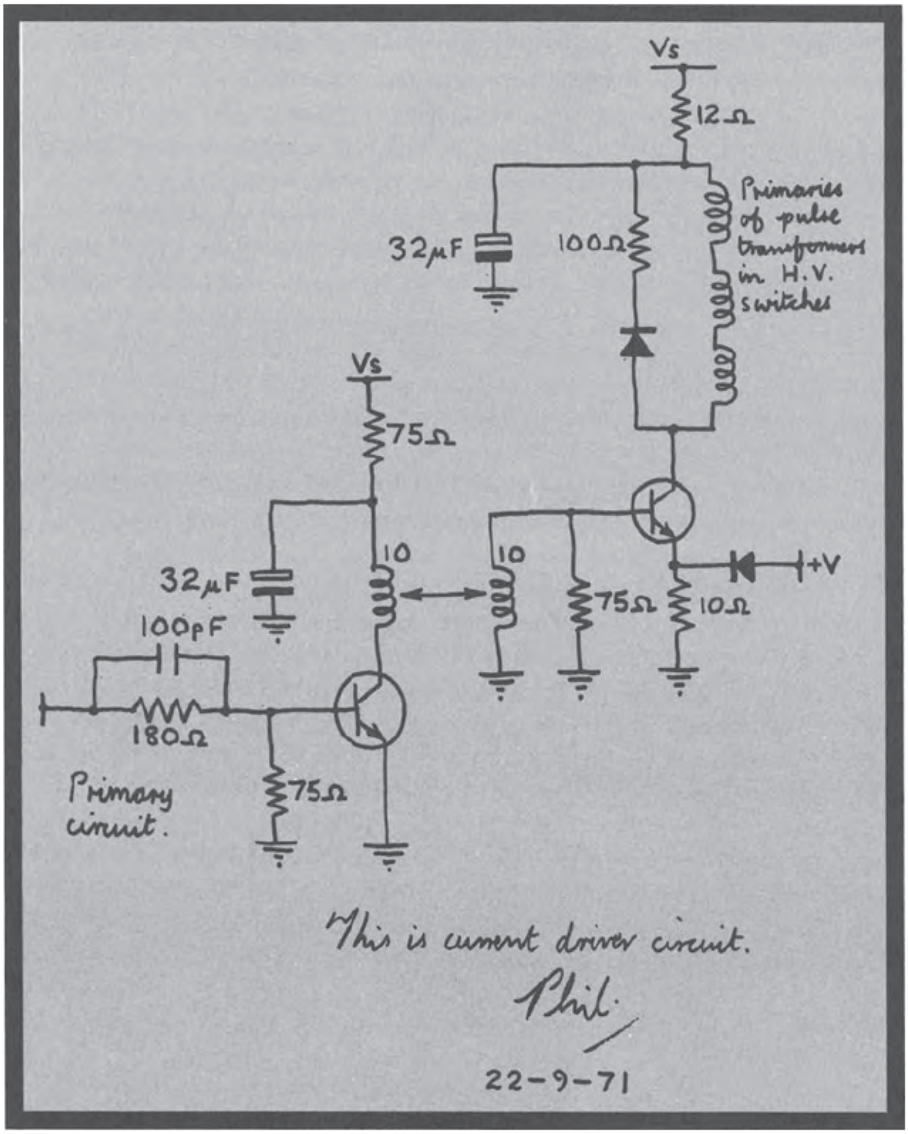


Fig. 5.7.8a Original CCITT image.

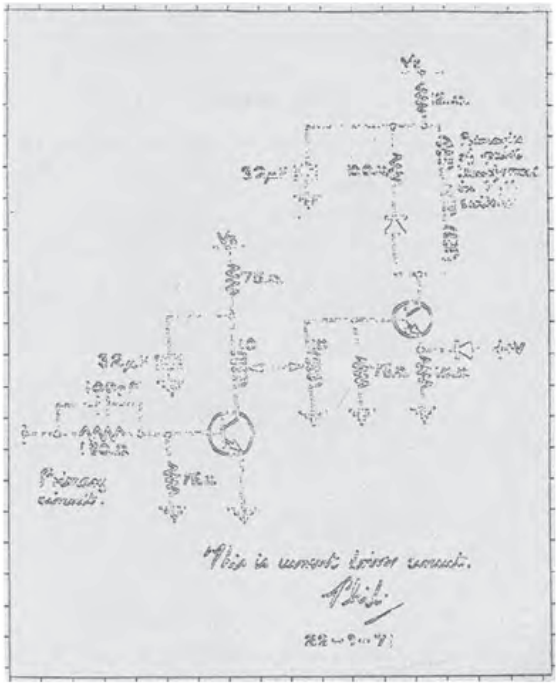


Fig. 5.7.8b Image of the prediction errors using the predictor of Fig. 5.7.4.

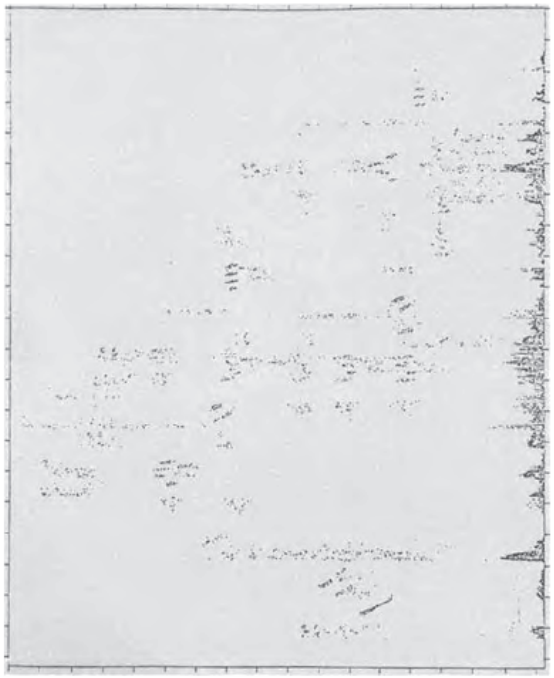


Fig. 5.7.8c The image of the reordered prediction errors. It is clear that the average run lengths here are larger than those in Fig. 5.7.8b.

CODING ALGORITHM	ENTROPY (BITS/PEL) CCITT IMAGE NUMBER							
	1	2	3	4	5	6	7	8
ONE-DIMENSIONAL RUN-LENGTH CODING	0.0505	0.0447	0.0914	0.1652	0.0988	0.0679	0.1791	0.087
RUN-LENGTH CODING OF PREDICTION ERRORS	0.0466	0.0373	0.0693	0.1640	0.0795	0.0482	0.1678	0.0671
RUN-LENGTH CODING OF ORDERED PREDICTION ERRORS; ONE SET OF CODES, GOODNESS THRESHOLD = 0.9	0.0390	0.0267	0.0571	0.1396	0.0652	0.0366	0.1400	0.0460
RUN-LENGTH CODING OF ORDERED PREDICTION ERRORS; TWO SETS OF CODES, GOODNESS THRESHOLD = 0.9;	0.0351	0.0233	0.0527	0.1282	0.0596	0 0355	0.1274	0.0419

Fig. 5.7.9 Comparative performance of Reordering Schemes for the eight CCITT Standard Images.

corresponding run-lengths of successive scan lines are transmitted. Fig. 5.7.10 shows the quantities Δ' and Δ'', and other messages such as new start, merge, etc., that are transmitted. Thus for each scan line, a transition element where the color changes from black to white (or white to black) is located and its location is transmitted with respect to a similar transition in the previous line. Several variations exist. For example, Fig. 5.7.10 shows transmission of white to black transitions (Δ') and black run-lengths (Δ''). Another variation of this principle is described in Chapter 6 as the CCITT standard two dimensional code. These schemes are essentially boundary coding schemes. The quantities Δ' and Δ'' indicate how the boundary propagates from one line to the next. We have seen in Chapter 3 that statistics of Δ' and Δ'' are highly nonuniform and, therefore, for higher efficiency, variable length codewords are used for representing Δ', Δ'', new start, and merge messages.

Schemes based on line-to-line run-length difference coding are quite efficient. Compression ratios of the same order as the best predictive scheme have been obtained. More details on the compression efficiency of a particular variation of this scheme (which has been chosen as a CCITT standard) are given in Chapter 6.

5.7.1d Block Coding

In block coding, a picture is divided into small rectangular blocks of $m \times n$ pels. Since each pel in this set of nm pels of the block can take on two values, there are 2^{nm} possible patterns of this block. However, not all these patterns are equally likely. Therefore, a variable length code is used to represent each of the patterns of the blocks. Optimum variable length codes based on Huffman construction become impractical for block sizes much larger than 3×3 (code table of size $2^9 = 512$). Therefore, much of the block coding literature is concerned with the assignment of suboptimum codes. One of these is indicated in Fig. 5.7.11. The code words for the most likely pattern (all zero or all white) is simply "0". The code words for the other patterns are obtained by the nm bits of the block preceded by the prefix "1". Thus, the code is a variable length code with only two possible lengths, namely 1 and $nm+1$. The compression ratio depends upon the size of the block and how many blocks consist of the all zero pattern. In general, block coding is less efficient than either predictive coding or line-to-line run-length differential coding.

5.7.1e Arithmetic Coding

We saw in Chapter 3 the basic principles of using arithmetic coding for bilevel images. First, we assumed the existence of a fairly good source model that could accurately estimate the conditional symbol probabilities given the recent past symbol values. Next, this source model was used to code the binary source symbols into a single real number x in the range 0 to 1. Fig. 3.1.10 showed an example of such coding for an initial four binary symbols.

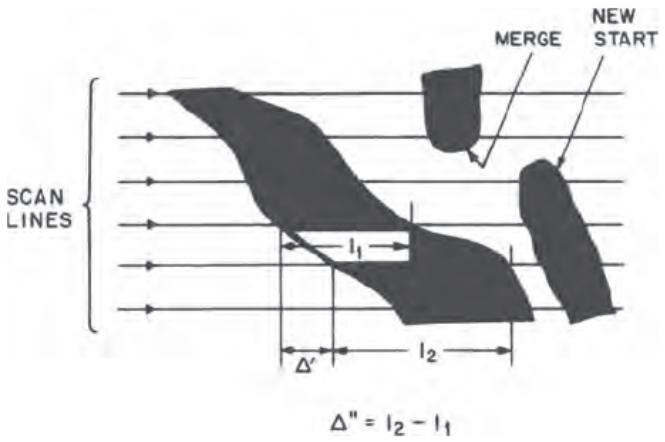


Fig. 5.7.10 Definition of transmitted quantities (e.g. Δ', Δ'') for line-to-line run-length difference coding.

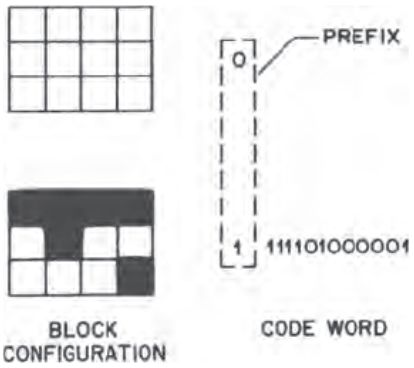


Fig. 5.7.11 Illustration of a suboptimum Block Code for a 4×3 block. The code for an all zero block is 0, since it occurs most often. There are 2^{12} codewords, only two of which are shown.

The portion of the unit interval on which x is known to lie after coding an initial sequence of symbols is known as the current coding interval. For each new binary input symbol the current coding interval is divided into two subintervals with sizes proportional to estimates of the conditional probabilities of symbol-value occurrences. The new current coding interval then becomes that portion of the old coding interval associated with the new symbol value actually occurring. Conceptually, this process continues until there are no more input symbols, after which a binary fraction representing some x within the last coding interval is transmitted.

Practically, of course, this process cannot be easily implemented as described, especially for long symbol strings. The accuracy required for the arithmetic simply become too large. However, certain approximations can be made to render the algorithm practical without seriously degrading performance. These will now be described.

By convention, the top subinterval of the partitioned current coding interval represents the Least Probable Symbol (LPS) probability, and the bottom subinterval the Most Probable Symbol (MPS) probability as shown in Fig. 5.7.12a. The size of the current coding interval is A. If p is the estimated probability of the LPS ($0<p<0.5$), then the size of the LPS subinterval is

$$LSZ = p \times A.$$

The current estimated value of x is C, which we take as the minimum value or *base* of the current coding interval. Before coding any symbols we set $C=0$, $A=1$.

Whenever an LPS is coded, the MPS subinterval $A-LSZ$ is added to C, and A is set to LSZ. Whenever an MPS is coded, the LPS subinterval LSZ is subtracted from A, and C is left unchanged. Thus, as coding proceeds, A always decreases while C either increases or stays the same.

In real implementations, of course, we can only store A and C values in registers having a finite number of bits. For A the problem is relatively easy. If after coding, A falls below 0.5 we successively *renormalize* by doubling both A and C until $A \geq 0.5$. Note that this may produce C values larger than 1. This is similar to floating point arithmetic.

The problem of finite accuracy storage of C values is not as easy to solve. What can be exploited, however, is the fact that as A decreases in value, there eventually becomes a time when some of the most significant bits of C stabilize to their final value and never change again. They can then be transmitted and no longer need to be stored within the encoder.

Thus, we partition the bit representation of C into four regions, as shown in Fig. 5.7.12b. The leftmost significant bits region is the Final (F) region for bits that have stabilized to their final value and are ready for transmission. The next region is the Tentative (T) region for bits that could still change, but will only do

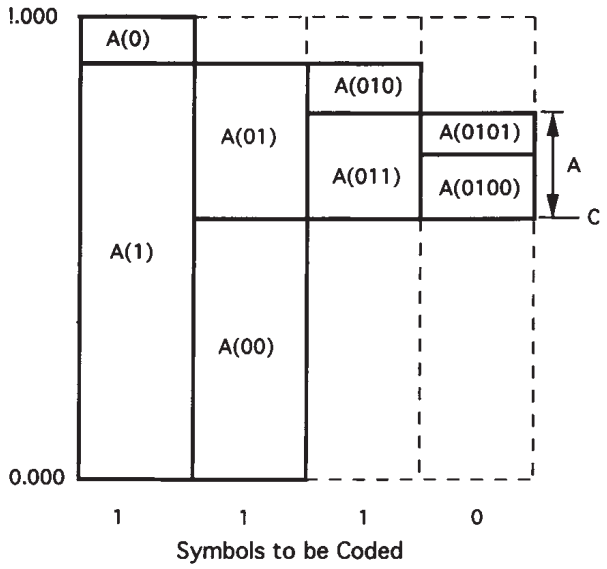


Fig. 5.7.12a Interval subdivision for Arithmetic Coding. The top subinterval represents the Least Probable Symbol (LPS), and the bottom subinterval represents the Most Probable Symbol (MPS).

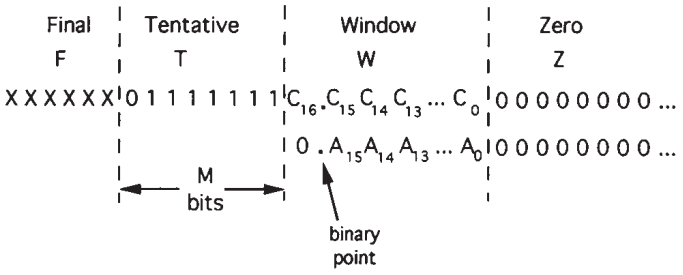


Fig. 5.7.12b Finite accuracy implementation of Arithmetic Coding. The current estimate C is partitioned into four regions - F, T, W and Z. Only the W region bits need be stored in the encoder.

so by a carry from the right. Next come the Window (W) region for bits that will change due to arithmetic operations, and finally the Zero (Z) region for bits that are always zero. The (floating) binary point is one bit from the left edge of the W region, as shown in the figure.

The W region is of fixed size, and for purposes of illustration we will assume 17 bits. The size of W also determines the minimum allowed value for *LSZ*, in our example 2^{-16}. Note that before coding any but the first symbol, $A_{16}=0$ and $A_{15}=1$.

A crucial property in this scenario is that before coding, the T region contains M bits, comprised of a zero on the left, followed by $M-1$ ones. Coding the next symbol then takes place as described above, with A decreasing and C either increasing (LPS) or staying the same (MPS). If coding causes a carry from W to T, then the T region changes to a one followed by $M-1$ zeros, and $C_{16}=0$. If this occurs, we then shift all the bits from the T region to the F region, and set $M=0$. Note this can only happen if an LPS is coded.

Following these coding operations, we then examine A_{15} to see if A has fallen below 0.5. If $A_{15}=0$ we renormalize. Note that coding an LPS always produces $A_{15}=0$.

Renormalization consists of two steps. First we examine C_{16}. If $C_{16}=0$ we shift all the bits, if any, from the T region to the F region, and set $M=0$. In the second step of renormalization, we shift the W region (and the floating binary point) one bit to the right and increment M by 1. The old C_{16} is thus shifted into the T region.

We continue renormalizing until $A_{15}=1$.

In the arrangement described above, the F region bits need to be stored within the encoder only until they are transmitted. Often one byte of storage is sufficient. All bits of the the W region need to be stored for the arithmetic operations. None of the bits for the Z region need to be stored.

The T region is a bit more complicated. Although none of the actual bits need to be stored, the value of M must be maintained. If a shift from T to F occurs due to coding, then a one followed by $M-1$ zeros must be shifted. If a shift occurs due to renormalization, then a zero followed by $M-1$ ones must be shifted.

After the last symbol is coded, all bits from the T and W regions can be shifted to the F region and transmitted.

Decoding is fairly straightforward. As soon as enough C bits arrive to indicate LSP or MPS, decode the symbol, recompute A and then renormalize if necessary. During renormalization, C bits to the left of the binary point can be discarded. Chapter 7 describes decoder implementation in more detail.

The compression performance of Arithmetic Coding is highly dependent on how good the probability estimates are. A probability model may exist from the start, or it may be developed on the fly as coding proceeds. In the latter case, typically, a running count is kept of symbol values conditioned on a

predetermined number of previous symbols. Much more will be said about Arithmetic Coding and probability modelling in Chapter 7.

5.7.2 Approximate Coding

The encoding techniques of the previous section allow perfect reproduction of the digitized picture at the receiver (in the absence of transmission error). Of course, the irreversible transformation of the image intensity that takes place due to sampling and thresholding makes it impossible to reproduce the analog intensity exactly. However, once thresholded, the two-level image can be reproduced at the receiver with no change if there are no transmission errors. In this section we look at methods by which irreversible or information-lossy processing of two-level pictures can be made to improve the compression ratio without significant degradation in the picture quality.

In general, the irreversible processing is performed before the final coding stage. Since most of the coding schemes of the previous section depend upon the ability to predict the color of a pel or the progression of a contour from line-to-line, irreversible processing techniques generally attempt to reduce the prediction error by preserving the continuity of the contours from line-to-line. We discuss two techniques below: One that is useful with predictive coding and a second that is useful with block coding.

The color of a picture element can always be changed so that the number of pels having nonzero prediction error are reduced. However, doing so can degrade the picture quality. Fig. 5.7.13a shows certain patterns in which the pel marked X may be changed in color without serious degradation in the overall picture quality. In pattern 1, pel X is regarded as an isolated pel due to noise whereas in patterns 2,3 and 4, pel X is located on a slant edge, and a change of its color results in rectangular edges. Fig. 5.7.13b shows the effect of such processing on two-dimensional prediction. After applying the two-dimensional prediction, the number of 1's in the prediction error signal is reduced. However, the picture quality in such a case obviously depends upon the patterns used for changing the color of pel X. No systematic investigation has yet been made to relate picture quality to patterns used in the approximation process and the resulting savings in bit-rate.

Picture approximation can also be used with block coding. The compression efficiency of a block code of the type given in the previous section depends to a large extent on the probability of occurrence of the all zero block, since the length of the codeword assigned to it is small. The compression efficiency can therefore be improved by increasing the probability of occurrence of the all zero block. For example, this may be done by changing all blocks having not more than K 1's to the all zero block. The amount of distortion introduced by such an approximation obviously depends upon the value of K used. Fig. 5.7.14 shows portions of two original digitized pictures. Approximated pictures using 4×4 blocks with $K=1$ and 2 are also shown in the

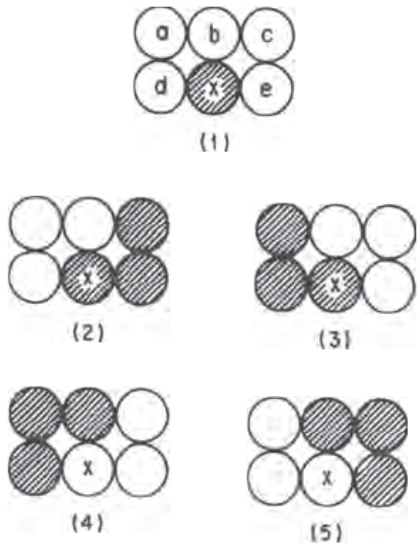


Fig. 5.7.13a Schemes for Modification of the pel *X* depending on the surrounding pattern.

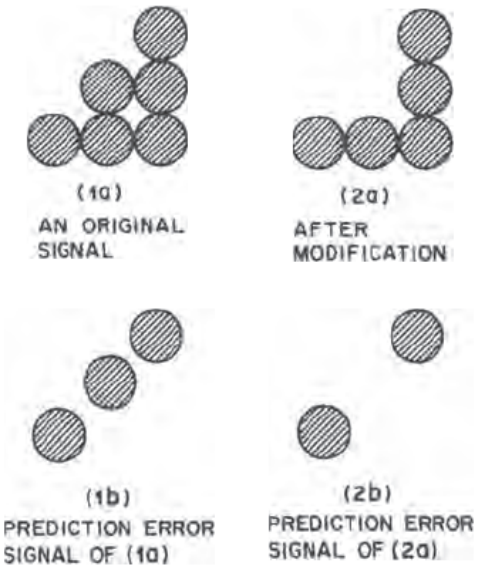


Fig. 5.7.13b An example showing the effectiveness of signal modification.

same figure. In general, the distortion introduced for $K = 1$ is not noticeable, but the compression ratio is improved only by about 8 percent. Distortions are noticeable for $K = 2$, and the compression ratio is improved by about 16 percent.

Another approximate block coding scheme is called *pattern matching*. This scheme is quite complex and still in its infancy. It is based on the observation that in documents containing text, several patterns (e.g., characters) occur repeatedly and, therefore, compression efficiency can be improved by identifying these patterns and transmitting to the receiver only their identification codes. First, a blocking operator is used to isolate patterns. Then, the first symbol encountered is placed in a library and its binary pattern is transmitted in a coded form. As each new pattern is detected, it is compared with entries of the library. If the comparison is within an error tolerance, the library identification code is transmitted along with the pattern location coordinates. Otherwise, the new pattern is placed in the library and its binary representation is transmitted in a coded form. Non-isolated patterns are left behind as residue, and are coded by conventional coding techniques. In one specific implementation based on this principle, a number of scan lines equal to about two to four times the average character height are stored in a scrolled buffer. This data is then examined line-by-line to determine if a black pel exists. If the entire line contains no black pels, this information is encoded by an end-of-line code. If a black pel exists, then a blocking operator is employed to isolate the pattern. For those isolated patterns, further processing is required to determine if a replica of the pattern under consideration already exists in the library. This process involves the extraction of a set of features, a screening operation to reject unpromising candidates, and finally a series of template matches. The first blocked pattern and its feature vector are put into the prototype library, and as each new blocked pattern is encountered, it is compared with each entry of the library that passes the screening test. If the comparison is successful, the library identification code is transmitted along with the location of the coordinates of the pattern. If the comparison is unsuccessful, the new pattern is both transmitted and placed in the library. Those areas of a document in which the blocker cannot isolate a valid pattern are assigned to a residue, and a conventional coding may be employed. The performance of the combined pattern matching scheme depends upon the matching algorithm and the tolerances used. In general, significant improvement is obtained for documents that are predominantly text (see Table 5.7.1). For other documents (e.g., containing drawings), the performance is about the same as predictive or line-to-line run-length difference coding. On the average, however, for the eight CCITT documents, the compression ratio of pattern matching[5.7.2] increases by almost a factor of two compared to the predictive or line-to-line run length difference coding.

Fig. 5.7.14 Approximate block coding. (a) original digitized image segments. Reconstructed pictures after block coding with $K = 1$ (b) and $K = 2$ (c).

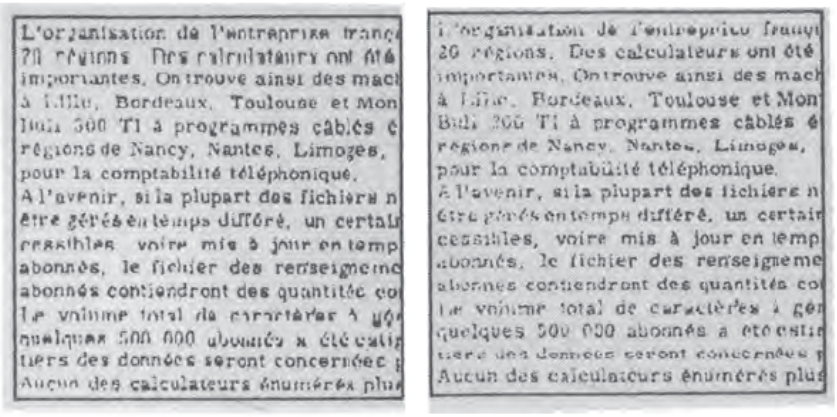


Fig. 5.7.15 Horizontal streaks caused by transmission errors in a picture coded with run-length codes. (a) original, (b) error image.

5.7.3 Transmission Errors

One penalty for efficient source coding is an increased sensitivity to errors in the transmission channel. This increased sensitivity is partially compensated for by the fact that after compression we have fewer bits per image and hence fewer chances for noise corruption. For example, at a resolution of about 8 pels/mm a standard CCITT image contains approximately 4 million bits. If we transmit such an image pel by pel through a channel with a bit error probability of 1 in 10^{-6}, we will get on the average 4 errors per image (assuming random errors rather than bursts). If an efficient coding scheme is used with a compression factor of 10, then we have 4 errors in every 10 images. The effect of the channel errors on the reproduced picture depends upon the encoding scheme and the technique used for combatting the channel errors.

In one-dimensional run-length coding algorithms a channel error will change a codeword so that the length of the corresponding run will be changed at the receiver. With variable length comma free codes, more information is lost because the decoder needs a certain interval to fall back into the correct codeword segmentation (i.e., regaining the synchronization). This causes a shift of the subsequent picture information. Fig. 5.7.15 shows an image that was coded by a one-dimensional run-length code and transmitted on a channel with average bit error rate of about 6×10^{-5}. The effect of a transmission error in this case is to cause horizontal streaks in the picture. These are generally limited to one scan line by introducing a synchronizing word at the end of each line. With two-dimensional codes, unless care is taken a single channel error can cause significant degradation in the rest of the picture.

In general, three methods have been considered to reduce the effects of transmission errors: a) Restriction of the damage caused by errors to as small an area as possible, b) Detection of errors and retransmission of corrupted blocks using an *Automatic Retransmission Request* (ARQ) System and c) Detection of errors and their correction at the receiver using a *Forward Acting Error Correcting* (FEC) code.

ARQ and FEC methods have not yet found wide acceptance since they are complex, add extra cost to the equipment and increase the transmission time. ARQ systems have the advantage of being very reliable and insensitive to changes in the channel error rate. Systems can be designed such that the probability of an undetected error in a block is less than 1 in 10^8. However, ARQ leads to reduction in the effective transmission rate. Also for channels that have inherently long transmission delay (e.g. satellite circuits) retransmission can become impractical. Forward error correcting codes, on the other hand, can have a higher effective transmission rate, but must be designed for the errors experienced on practical channels. Thus, they can be sensitive to changes in the error patterns.

In the case of one-dimensional run-length coding, several techniques exist to improve the quality of reconstructed pictures in the presence of transmission

errors. The detection of a channel error can be easily performed by summing up the decoded run-lengths between two synchronizing words and comparing the result with the nominal length of a scan line. If an error is detected, one of the following error concealment techniques can be used to improve the reconstructed picture quality: a) replace the damaged line by an all white line, b) repeat the previous line, c) print the damaged line, d) use a line-to-line processing or correlation technique to reconstruct as much of the line as possible. Picture quality improvement can be rather high for (b). Correlation methods (d) take advantage of the fact that the error recovery period is often short, and thus, they attempt to retain as much as possible of the correctly decoded data on a damaged line. This is achieved by attempting to locate the damaged run-lengths. One method is to measure the correlation between group of pels on the damaged line with corresponding groups on the adjacent lines above and below. Where the correlation is good, generally at the beginning and end of the damaged scan line, the scan line data is used to reconstruct the line. The part of the scan line that is assumed to be damaged is then replaced by a corresponding part of the previous line. This method works well most of the time, but does result in a loss of vertical resolution.

In the case of two-dimensional coding, in order to prevent the vertical propagation of damage caused by transmission errors, no more than $K-1$ successive lines are two-dimensionally coded. The next line is one-dimensionally coded. Usually K is set equal to two at 4 pels/mm and set equal to four at 8 pels/mm resolution. Any transmission error thus will be localized to a group of K lines in the picture.

5.7.4 Soft Facsimile

Cathode ray tubes (CRT) and other erasable display media have proliferated in recent years. Also, the cost of refresh frame buffers has decreased dramatically in the past few years. This has given impetus to new techniques for transmission of still pictures. Traditionally, transmission of still pictures has used the regular scanning pattern, i.e., picture elements are transmitted sequentially along the scanning direction (from left to right and top to bottom). This scanning pattern cannot be easily changed when paper output is used. However, for low capacity channels, it is often more desirable to transmit and display the pictures in a progressive (or stage-by-stage) fashion. This is done by transmitting a crude representation of the image first, and then adding details later. This is possible since the contents of the refresh frame memory used at the receiver (to provide a flicker free display) can be changed in almost any random order, and not necessarily in a regular line-by-line scan. Progressive presentation is desirable where a viewer wishes to recognize image content as soon as possible, and perhaps halt the transmission of unnecessary detail, or initiate actions simultaneous to transmission of the remaining detail. Also, besides providing an aesthetically pleasing sequence that holds the viewer's

attention, progressive presentation also allows more efficient browsing through a picture data base.

The usefulness of such progressive presentation methods is somewhat speculative at this time. Also, new techniques that are being proposed for hierarchical representation of pictures and subsequent coding are still in their infancy, and significant experience with them is lacking. For this reason, we give below a brief summary of some techniques that have been tried. The reader is referred to references for details. Fig. 5.7.16 shows results of computer simulation of one specific technique. Progressive transmission, as well as the top to bottom regular scanning method are shown. For each case, development of the pictures at various transmission times is shown. It is clear from the photographs that a quicker recognition of the picture may often be possible by progressive transmission.

Division of a picture into several subpictures (corresponding to the stages of progressive transmission) by various subsampling patterns can also be used for progressive transmission. In this case, the first stage may correspond to a known subsampling pattern, and the pels of this stage may be coded by one of the many standard techniques. At the receiver, the pels that are not transmitted in the first stage, are reproduced by interpolation. The quality of the pictures produced in different stages is a function of the subsampling pattern as well as the interpolation techniques. Transform coding methods can also be used for progressive transmission especially for multilevel pictures. This is easily done by transmitting sequentially one coefficient for all the blocks in an image and then transmitting the remaining coefficients in some sequence that is known to the receiver. The quality of the pictures will obviously depend upon the order of coefficient transmission. It is not clear whether lower order coefficients (which give low frequency representation) should be sent before the higher order coefficients (containing predominantly edge information). The problem is further complicated by the fact that different coefficients require different accuracy. In any case, transform coding methods apply more readily to progressive transmission of gray level images than to two-level images.

Another block-by-block method is to form a hierarchical structure of successively smaller sub divisions of an image, and to transmit the higher level data first. These can be applied to gray level as well as two-level images. Another method, primarily suitable for two-level signals, is based on the *growth geometry* technique. In this method, within one image, aggregates of pels are identified that are capable of being generated by applying specified growth processes to particular pels, called seed pels. The data transmitted is the location, type of growth, and the number of growth stages for each seed pel. At the receiver, pels in the neighborhood of each seed pel are attached according to the indicated growth process, and thus regenerate the image exactly. It is difficult to compare these techniques primarily because there is no good objective measurement of human-display interaction or the quality of browsing.

Fig. 5.7.16 Development of an image in a progressive transmission scheme and its comparison with a sequential top-to-bottom scheme (called slow-scan facsimile).

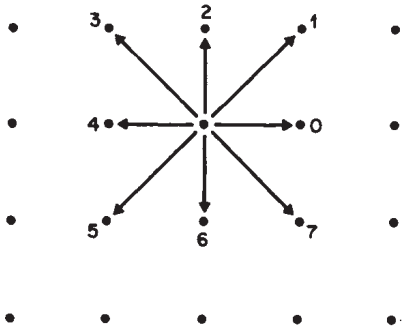


Fig. 5.7.17 Definition of Chain Code for line drawings. Eight directions shown by arrows are encoded by a three bit word. Dots represent the sampling grid.

Table 5.7.1 Comparison of compression ratios with pattern matching and the modified read code recommended by the CCITT (see Chapter 6 for more details).

TEST DOCUMENT	PATTERN MATCHING	MODIFIED READ
CCITT1	62.86	35.28
CCITT2	46.21	47.51
CCITT3	26.00	17.88
CCITT4	35.36	7.41
CCITT5	29.16	15.93
CCITT6	28.85	30.82
CCITT7	22.60	26.87
CCITT8	29.19	26.87
AVERAGE	29.19	15.54

Table 5.7.2 A Variable Length Code for Differential Chain of a Line Drawing.

DIRECTION OF THE DIFFERENTIAL CHAIN	CODEWORD
0	0
+1	01
-1	011
+2	0111
-2	01111
+3	011111
-3	0111111
+4	01111111
-4	011111111

5.7.5 Coding of Line Drawings

As we saw in Section 1.9, line drawings are a special subclass of two-level photographic images in which the picture consists essentially of interconnections of thin lines on a contrasting background. The thickness and color of the lines as well as the color and texture of the background are of little or, at most, symbolic significance. Such line drawings can be represented (see Fig. 1.9.5) on a rectangular grid by a sequence of pels with their corresponding (x,y) coordinates. However, specification of the picture by (x,y) coordinates of the pels is inefficient. A more efficient representation is by *chain codes* in which the vector joining the two successive pels is assigned a codeword. For a rectangular grid, since there are eight neighbors of any pel (see Fig. 5.7.17), only eight directions for this vector need to be specified, which requires 3 bits. More efficiency is obtained by *differential chain coding* in which each pel is represented by the difference between two successive absolute codes. Although, there are still eight codewords, in practice, they are not found to be equally likely, and therefore a variable length code of the form shown in Table 5.7.2 can be used. Simulation results show that such a code requires typically 1.8 to 1.9 bits/pel for alphanumeric characters and about 1.5 to 1.9 bits/pel for thinned edges of objects.

Appendix - Fast DCT

Here we provide fast 1D DCT and IDCT algorithms for blocksize 8. They are based on the theory of rotations[5.3.7a] and minimize the number of multiplications.

```
                /* 1D  DCT  block size 8*/
                /*Computed by rotations as in Reference [5.3.7a] */
#define  PI     3.14159265358979323846264338327950288419716 9
#include <math.h>
dct(pels,coefs)
short   pels[8];        /* input values */
double  coefs[8];       /* output values */
{
        static int i;
        static double   c0,c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12;
        static double   alpha;
        static double   ts,m1,m2,m3;            /* temp variables */
        static double   s1[8], s2[8], s3[8], s4[8], s5[8];
        static int first=1;
        double cos(),sin(),sqrt();
```

```
if(first){
   first=0;
   alpha = 2*PI/(4*8)        ;
   c0  =  sin(4.0*alpha) - cos(4.0*alpha);
   c1  =  cos(4.0*alpha);
   c2  = - (sin(4.0*alpha) + cos(4.0*alpha));
   c3  =  sin(1.0*alpha) - cos(1.0*alpha);
   c4  =  cos(1.0*alpha);
   c5  = - (sin(1.0*alpha) + cos(1.0*alpha));
   c6  =  sin(2.0*alpha) - cos(2.0*alpha);
   c7  =  cos(2.0*alpha);
   c8  = - (sin(2.0*alpha) + cos(2.0*alpha));
   c9  =  sin(3.0*alpha) - cos(3.0*alpha);
   c10 =   cos(3.0*alpha);
   c11 = - (sin(3.0*alpha) + cos(3.0*alpha));
   c12 =   (double)1 / sqrt((double)2);
}
s1[0] = pels[0] + pels[7]; s1[1] = pels[0] - pels[7];
s1[2] = pels[2] + pels[1]; s1[3] = pels[2] - pels[1];
s1[4] = pels[4] + pels[3]; s1[5] = pels[4] - pels[3];
s1[6] = pels[5] + pels[6]; s1[7] = pels[5] - pels[6];

s2[0] = s1[1]; s2[1] = s1[5];
s2[2] = s1[0] + s1[4]; s2[3] = s1[0] - s1[4];
s2[4] = s1[2] + s1[6]; s2[5] = s1[2] - s1[6];
s2[6] = s1[3] + s1[7]; s2[7] = s1[3] - s1[7];

s3[0] = s2[0]; s3[1] = s2[1];
s3[2] = s2[2]; s3[3] = s2[5]*c12;
s3[4] = s2[3]; s3[5] = s2[4];
s3[6] = s2[6]; s3[7] = s2[7]*c12;

/*The following can be skipped and incorporated
  into the quantization process by doubling
  the quantizer step size                    */
for (i=0; i<8; ++i) s3[i] = s3[i]/2.0;

s4[0] = s3[2]; s4[1] = s3[4];
s4[2] = s3[5]; s4[3] = s3[6];
s4[4] = s3[0] + s3[3]; s4[5] = s3[0] - s3[3];
s4[6] = s3[7] + s3[1]; s4[7] = s3[7] - s3[1];
```

```
        ts = s4[0] + s4[2]; m1 = s4[0] * c0; m2 = ts * c1;
        m3 = s4[2] * c2; s5[0] = m2 + m3; s5[1] = m1 + m2;

        ts = s4[4] + s4[6]; m1 = s4[4] * c3; m2 = ts * c4;
        m3 = s4[6] * c5; s5[2] = m2 + m3; s5[3] = m1 + m2;

        ts = s4[1] + s4[3]; m1 = s4[1] * c6; m2 = ts * c7;
        m3 = s4[3] * c8; s5[4] = m2 + m3; s5[5] = m1 + m2;

        ts = s4[5] + s4[7]; m1 = s4[5] * c9; m2 = ts * c10;
        m3 = s4[7] * c11; s5[6] = m2 + m3; s5[7] = m1 + m2;

        coefs[0] = s5[1]; coefs[1] = s5[2]; coefs[2] = s5[4];
        coefs[3] = s5[6]; coefs[4] = s5[0]; coefs[5] = s5[7];
        coefs[6] = s5[5]; coefs[7] = s5[3];
return(0);
}
                        /*1d idct blocksize 8*/
#define PI      3.14159265358979323846264338327950288419716939
#include <math.h>

idct(coefs,pels)
double coefs[8];
short pels[8];
{
        static double dout[8];
        static double     c1,c2,c3,c4,c5,c6,c7,c8,c9;
        static double     s1[8],s2[8],s3[8],s4[8],s5[8];
        static double     nmax2, alpha;
        static int i;
        static int first=1;

        if(first){
          first=0;
          nmax2 = 4;
          alpha = 2*PI/(4*8)      ;

          c1 = cos(nmax2*alpha); c2 = sin(nmax2*alpha);
          c3 = cos(alpha); c4 = sin(alpha);
          c5 = cos(2*alpha); c6 = sin(2*alpha);
          c7 = cos(3*alpha); c8 = sin(3*alpha);
          c9 = sqrt((double)2);
        }
```

```
        s1[0] = coefs[4]; s1[1] = coefs[0]; s1[2] = coefs[1];
        s1[3] = coefs[7]; s1[4] = coefs[2]; s1[5] = coefs[6];
        s1[6] = coefs[3]; s1[7] = coefs[5];

        s2[0] = c1*2*s1[0] + c2*2*s1[1]; s2[1] = c5*2*s1[4] + c6*2*s1[5];
        s2[2] = c1*2*s1[1] - c2*2*s1[0]; s2[3] = c5*2*s1[5] - c6*2*s1[4];
        s2[4] = c3*2*s1[2] + c4*2*s1[3]; s2[5] = c7*2*s1[6] + c8*2*s1[7];
        s2[6] = c3*2*s1[3] - c4*2*s1[2]; s2[7] = c7*2*s1[7] - c8*2*s1[6];

        s3[0] = 0.5*s2[4] + 0.5*s2[5]; s3[1] = 0.5*s2[6] - 0.5*s2[7];
        s3[2] = s2[0]; s3[3] = 0.5*s2[4] - 0.5*s2[5];
        s3[4] = s2[1]; s3[5] = s2[2]; s3[6] = s2[3];
        s3[7] = 0.5*s2[6] + 0.5*s2[7];

        s4[0] = s3[0]; s4[1] = s3[1]; s4[2] = s3[2];
        s4[3] = s3[4]; s4[4] = s3[5]; s4[5] = s3[3]*c9;
        s4[6] = s3[6]; s4[7] = s3[7]*c9;

        s5[0] = 0.5*s4[2] + 0.5*s4[3]; s5[1] = s4[0];
        s5[2] = 0.5*s4[4] + 0.5*s4[5]; s5[3] = 0.5*s4[6] + 0.5*s4[7];
        s5[4] = 0.5*s4[2] - 0.5*s4[3]; s5[5] = s4[1];
        s5[6] = 0.5*s4[4] - 0.5*s4[5]; s5[7] = 0.5*s4[6] - 0.5*s4[7];

        dout[0] = 0.5*s5[0] + 0.5*s5[1]; dout[1] = 0.5*s5[2] - 0.5*s5[3];
        dout[2] = 0.5*s5[2] + 0.5*s5[3]; dout[3] = 0.5*s5[4] - 0.5*s5[5];
        dout[4] = 0.5*s5[4] + 0.5*s5[5]; dout[5] = 0.5*s5[6] + 0.5*s5[7];
        dout[6] = 0.5*s5[6] - 0.5*s5[7]; dout[7] = 0.5*s5[0] - 0.5*s5[1];

        for(i=0; i<=7; i++){
                if( dout[i]<0.0 )pels[i] = dout[i] - 0.5;
                else                    pels[i] = dout[i] + 0.5;
        }
        return(0);
}
```

References

5.1.1 B. M. Oliver, J. R. Pierce and C. E. Shannon, "The Philosophy of PCM", *Proceedings of IRE*, Vol. 36, Oct. 1948, pp. 1324-1331.

5.1.2 W. M. Goodall, "Television by Pulse Code Modulation", *Bell System Technical Journal*, Vol. 30, Jan. 1951, pp. 33-49.

5.1.3 L. G. Roberts, "Picture Coding Using Pseudo-Random Noise", *IRE Trans. on Information Theory*, Vol. IT-8, February 1962, pp. 145-154.

5.1.4 A. K. Bhushan, "Efficient Transmission and Coding of Color Components", M.S. Thesis, Massachusetts Institute of Technology, Cambridge, MA, June 1977.

5.1.5 W. Frei, P. A. Jaeger and P. A. Probst, "Quantization of Pictorial Color Information", *Nachrichtentech Z*, Vol 61, 1972, pp. 401-404.

5.1.6 B. Marti, "Preliminary Processing of Color Images", CCETT ATA/T/3/73, September 5, 1973.

5.1.7 L. Stenger, "Quantization of TV Chrominance Signals Considering the Visibility of Small Color Differences", *IEEE Trans. on Communications*, Vol. COM-25, No. 11, November 1977.

5.2.1 C. C. Cutler, "Differential Quantization of Communication Signals", U.S. Patent 2 605 361, July 1952.

5.2.2 F. deJager, "Delta Modulation, A Method of PCM Transmission using a 7-Unit Code", Philips Research Reports, Dec. 1952, pp. 442-466.

5.2.3 C. W. Harrison, "Experiments with Linear Prediction in Television", *Bell System Technical Journal*, Vol. 31, July 1952, pp. 764-783.

5.2.4 T. Kailath, *Linear Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

5.2.5 A. Habibi, "Comparison of Nth order DPCM encoder with Linear Transformations and Block Quantization Techniques", *IEEE Trans. on Communication Technology*, Vol. COM-19, Dec. 1971, pp. 948-956.

5.2.6 H. G. Musmann, "Predictive Coding" Chapter in the book *Image Transmission Techniques*, edited by W. K. Pratt, Academic Press 1979.

5.2.7 R. E. Graham, "Predictive Quantizing of Television Signals", *IRE Wescon Convention Record*, Vol. 2, pt 4, 1958, pp. 147-157.

5.2.8 B. G. Haskell, "Entropy Measurements for Nonadaptive and Adaptive, Frame-to-Frame, Linear Predictive Coding of Video Telephone Signals", *Bell System Technical Journal*, Vol. 54, No. 6, August 1975, pp. 1155-1174.

5.2.9 H. G. Musmann, P. Pirsch, and H.-J. Grallert, "Advances in Picture Coding", *Proceedings of IEEE*, April 1985.

5.2.10 J. O. Limb, and H. A. Murphy, "Measuring the Speed of Moving Objects from Television Signals", *IEEE Trans. on Comm.*, April 1975.

5.2.11 A. N. Netravali and J. D. Robbins, "Motion-Compensated Television Coding: Part I", *Bell System Technical Journal*, Vol. 58, No. 33, March 1979, pp. 631-669.

5.2.12 A. N. Netravali and J. D. Robbins, "Motion Compensated Coding: Some New Results", *Bell System Technical Journal*, Nov. 1980.

5.2.13 J. A. Stuller, A. N. Netravali and J. D. Robbins, "Interframe Television Coding Using Gain and Displacement Compensation", *Bell System Technical Journal*, Sept. 1980.

5.2.14 K. A. Prabhu and A. N. Netravali, "Motion Compensated Components Color Coding," *IEEE Trans. Comm.*, Dec. 1982.

5.2.15 K. A. Prabhu and A. N. Netravali, "Motion Compensated Composite Color Coding", *IEEE Trans. Comm.*, Feb. 1983.

5.2.16 D. J. Connor, R. C. Brainard and J. O. Limb, "Intraframe Coding for Picture Transmission", *IEEE Proceedings*, July 1972.

5.2.17 R. Jung and R. Lippman, "Error Response of DPCM Decoders", *Sonderdruck aus Nachrichtentechnische Zeitschrift*, Bd. 28, 1974, pp. 431-436.

5.2.18 J. Max, "Quantizing for Minimum Distortion", *IEEE Trans. on Information Theory*, Vol. IT-6, March 1960, pp. 7-12. Also, S. P. Lloyd," Least Squares Quantization in PCM", *IEEE Trans on Information Theory*, March 1982, pp. 129-136.

5.2.19 D. K. Sharma and A. N. Netravali, "Design of Quantizers for DPCM Coding of Picture Signals", *IEEE Trans. on Communication*, Vol. COM-25, Nov. 1977, pp. 1267-1274.

5.2.20 A. N. Netravali, "On Quantizers for DPCM Coding of Picture Signals", *IEEE Trans. on Information Theory*, Vol. IT-23, No. 3, May 1977, pp. 360-370.

5.2.21 A. N. Netravali and C. B. Rubinstein, "Quantization of Color Signals", *Proceedings of IEEE*, Vol. 65, No. 3, August 1977, pp. 1177-1187.

5.2.22 A. N. Netravali and B. Prasada, "Adaptive Quantization of Picture Signals Using Spatial Masking", *Proceedings of IEEE*, Vol. 65, April 1977, pp. 536-548.

5.2.23 P. Pirsch, "Block Coding of Color Video Signals", in Proceedings of National Telecommunications Conference, 1977, pp. 10.5.1-10.5.5.

5.3.1 E. O. Brigham, *The Fast Fourier Transform*, Prentice-Hall, Englewood Cliffs, N.J., 1984.

5.3.2 *Programs for Digital Signal Processing*, Edited by IEEE Acoustics, Speech and Signal Processing Society, IEEE Press, New York, 1979.

5.3.3 H. F. Silverman, "An Introduction to Programming the Winograd Fourier Transform Algorithm (WFTA)," *IEEE Trans. on Acoustics, Speech and Signal Processing*, v. ASSP-25, No. 2, April 1977, pp. 152-165.

5.3.4 W. K. Pratt, J. Kane and H. C. Andrews, "Hadmard Transform Image Coding," *Proc. IEEE*, v. 57, No. 1, Jan. 1969, pp. 58-68.

5.3.5 R. D. Brown, "A Recursive Algorithm for Sequency Ordered Fast Walsh Transforms," *IEEE Trans. Computers*, v. C-26, No. 8, Aug. 1977, pp. 819-822.

5.3.6 N. Ahmed and K. R. Rao, *Orthogonal Transform for Digital Signal Processing*, Springer-Verlag, New York, 1975.

5.3.7 J. Makhoul, "A Fast Cosine Transform in One and Two Dimensions," *IEEE Trans. Acoustics, Speech and Signal Processing*, v. ASSP-28, No. 1, Feb. 1980, pp. 27-34.

5.3.7a C. Loeffler, A. Ligtenberg and G. S. Moschytz, "Practical fast 1D DCT algorithm with 11 multiplications," *Proceedings IEEE ICASSP-89*, v.2, pp. 988-991, Feb. 1989.

5.3.8 B. Fino, "An Experimental Study of Image Coding Utilizing the Haar Transform and the Hadamard Complex Transform," *Ann. Telecommunications*, v. 27 (5-6) pp. 185-208, 1972 (in French).

5.3.9 W. K. Pratt, W. Chen and L. R. Welch, "Slant Transform Image Coding," *IEEE Trans. Communications,* V. COM-22, pp. 1075-1093, Aug. 1974.

5.3.10 A. K. Jain, "Image Data Compression: A Review," *Proc. IEEE,* v. 69, No. 3, pp. 349-389, March 1981.

5.3.11 H. C. Andrews and C. L. Patterson, "Outer Product Expansions and Their Uses in Digital Image Processing," *IEEE Trans. Computers,* v. C-25, No. 2 pp. 140-148, Feb. 1976.

5.3.12 F. W. Mounts, A. N. Netravali and B. Prasada, "Design of Quantizers for Real-Time Hadamard Transform Coding of Pictures," *Bell Sys. Tech. J.,* v. 56, No. 1, January 1977, pp. 21-48.

5.3.13 A. K. Jain and S. H. Wang, "Stochastic Image Models and Hybrid Coding," Final Report Contract No. N00953-77-C-003 MJE, Dept. of Elec. Engin, State U. of New York at Buffalo, October 1977.

5.3.14 J. Huang and P. Schultheiss, "Block Quantization of Correlated Gaussian Random Variables," IEEE Trans. Communications, COM-11, 1963, pp. 286-296.

5.3.15 W. H. Chen and C. N. Smith, "Adaptive Coding of Monochrome and Color Images," *IEEE Trans. Communications,* v. COM-25, No. 11, November 1977, pp. 1285-1292.

5.3.16 J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering,* John Wiley & Sons, Inc., New York, 1965.

5.3.17 J. O. Limb, C. B. Rubinstein and J. E. Thompson, "Digital Coding of Color Video Signals - A Review," *IEEE Trans. Communications,* v COM-25, No. 11, November 1977, pp. 1349-1385.

5.3.18 T. R. Natarajan and N. Ahmed, "On Interframe Transform Coding," *IEEE Trans. Communications,* v. COM-25, No. 11, November 1977, pp. 1323-1329.

5.3.19 J. R. Jain and A. K. Jain, "Interframe Adaptive Data Compression Techniques for Images," Signal and Image Processing Laboratory—Dept. of Elec. and Computer Engin., University of Calif., Davis, CA., AD-A078841.

5.3.20 S. C. Knauer, "Real-Time Video Compression Algorithm for Hadamard Transform Processing," *IEEE Trans. Electromagnetic Compatibility,* v. EMC-18, No. 1, February 1976, pp. 28-36.

5.3.21 A. G. Tescher, "Transform Image Coding," Chapter 4 of *Image Transmission Techniques,* W. K. Pratt ed., Academic Press, New York, 1979.

5.4.1 A. Habibi, "Hybrid Coding of Pictorial Data," *IEEE Trans. Communications,* v. COM-22, No. 5, May 1974, pp. 614-624.

5.4.2 J. A. Roese, "Hybrid Transform/Predictive Image Coding," Chapter 5 of *Image Transmission Techniques,* W. K. Pratt ed., Academic Press, New York, 1979.

5.4.3 R. J. Clarke, "Hybrid Intraframe Transform Coding of Image Data," *IEE Proc.,* v. 131 part F, No. 1, Feb. 1984, pp. 2-6.

5.5.1 A. Gersho and R. Gray, *Vector Quantization and Signal Compression,* Kluwer Academic Publishers, 1992.

5.5.2 Y. Linde, A. Buzo and R. Gray, "An Algorithm for Vector Quantizer Designs," *IEEE Trans Commun.,* COM-28, Jan. 1980, pp. 84-95.

5.5.3 W. H. Equitz, "Fast Algorithms for Vector Quantization Picture Coding," M. S. Thesis, MIT, June 1984. Also see "Some Methods for Classification and Analysis of Multivariate Observations", MacQueen, Proc. 5th Berkeley Symp. on Math, Statistics and Probability, v. 1, pp. 281-296, 1967.

5.5.4 A. Gersho and B. Ramamurthi, "Image Coding using Vector Quantization," Proc.of Int. Conf. ASSP, Paris, 1982, pp. 428-431.

5.5.5 D. J. Healy and D. R. Mitchell, "Digital Video Bandwidth Compression Using Block Truncation Coding", *IEEE Trans. Commun.*, COM-29, Dec. 1981, pp. 1809-1823.

5.6.1 P. J. Burt and E. H. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans. on Communications*, COM-31, April 1983, pp. 532-540.

5.6.2 J. W. Woods and Sean D. O'Neil, "Subband Coding of Images," *IEEE Trans. on Acoustics, Speech and Signal Proc.*, ASSP-34, Oct. 1986, pp. 1278-1288.

5.7.1 R. Hunter and A. H. Robinson, "International Digital Facsimile Coding Standards," *Proceedings of IEEE*, July 1980.

5.7.2 O. Johnsen, J. Segen and G. L. Cash, "Coding of Two-Level Pictures of Pattern Matching and Substitution," *Bell System Technical J.*, Vol. 62, Oct. 1983, pp. 2513-2545.

Questions for Understanding

5.1 What is the quantization noise PSNR in dB for 7 bit, 8 bit and 9 bit PCM? What are the impairments for each on the CCIR 5-point scale?

5.2 Explain the difference between 1-bit PCM and DM. Which is better and why?

5.3 Explain granular noise and slope overload. How does compand DM improve these impairments?

5.4 What are instantaneous and syllabic DM companding?

5.5 What are the advantages and disadvantages of minimum MSE linear predictor in image coding?

5.6 What factors should be considered in choosing linear predictor weighting coefficients?

5.7 Explain the operation of adaptive intrafield predictive coders.

5.8 What are the advantages and disadvantages of interframe predictive coding?

5.9 What are the underlying assumptions of block matching and pel recursive motion estimation?

5.10 When would gain compensation be useful?

5.11 What methods are used in predictive coding to deal with transmission bit errors?

5.12 What causes edge busyness?

5.13 What are mid-step and mid-riser quantizers? When would each be used?

5.14 How are visibility thresholds used to design quantizers?

5.15 What are the relative advantages and disadvantages of uniform and companded quantization in predictive coding? Also, variable length coding and fixed length coding?

5.16 What are the differences between MMSE and MSSDE quantizers?

5.17 Explain reflected quantizers.

5.18 What are masking functions and how are they used in adaptive quantizers for predictive coding? What are the advantages and disadvantages of this approach?

5.19 What are the advantages of separable transform coding?

5.20 Explain the relative merits of RDFT, WHT, DCT and KLT.

5.21 How is quantization different for DPCM and transform coding?

5.22 Explain quantization error and truncation error in optimum transform coding.

5.23 Derive Eq. (5.3.86).

5.24 Explain adaptive transform coding with classes.

5.25 What are threshold coding and zonal coding?

5.26 What are the advantages and disadvantages of 3D transform coding?

5.27 Explain the merits of intra and inter frame hybrid transform coding.

5.28 Explain motion compensated hybrid transform coding.

5.29 How is vector quantization different than transform coding?

5.30 Explain the LBG algorithm for VQ. What are the difficulties of using it?

5.31 What is multistage VQ?

5.32 Explain BTC.

5.33 What is interpolative coding? How is motion adaptation used?

5.34 Explain pyramid coding. Also, subband coding.

5.35 What is run length coding?

5.36 Explain reordering, as applied to graphics coding.

5.37 Explain line-to-line run-length difference coding.

5.38 What are the merits and demerits of approximate graphics coding?

5.39 What is pattern matching coding?

5.40 What are ARQ and FEC?

5.41 Explain the chain code algorithm.

6

Examples of Codec Designs

In this chapter, we describe several image codec designs in some detail. Our purpose is not to present the current state of the art, for that changes almost on a daily basis. Rather, we wish to show how the basic principles discussed in previous chapters might be applied in an interrelated way to practical systems. Some applications require a simple codec without a high degree of data compression. In other cases we need a low bit-rate because of expensive transmission channels, and codec cost is less of a consideration. In still other situations we want very high picture quality, with less concern about low complexity and compression capabilities. The techniques described in this chapter cover a range of complexity, picture quality and data compression in order to assist in the making of sound engineering decisions when designing image communication systems. However, some of the implementations also contain a certain amount of ad hoc procedures that can only be optimized by experimentation.

We start (see Fig. 6.0.1) by describing a fairly simple intrafield DPCM codec for composite NTSC color television. This design has been implemented in VLSI and provides a good quality videoconferencing picture. Next, we discuss an interfield adaptive DPCM coding technique applied to broadcast quality, NTSC color television. The methodology of this codec is also applicable to component color and monochrome pictures. However, in these cases the bit-rates, predictors and quantizers will be somewhat different. Next, we describe an interfield hybrid transform coder for component color TV. This approach, in a modified form, is also applicable to single color images. Section 6.5 describes a scene decomposition algorithm for coding single images. The next section describes interframe coders suitable for videoconferencing, and the final section discusses two-level graphics coding.

Section	Image Source	Principal Technique	Uncoded Bit-Rate	Coded Bit-Rate	Complexity	Image Quality
6.1	NTSC Color Video	Intrafield DPCM	86 Mbits/s	43 Mbits/s	Low	4
6.2	NTSC Color Video	Adaptive Interfield DPCM	129 Mbits/s	42 Mbits/s	Moderate	5
6.3	Component Color Video	Interfield Hybrid Transform	64 Mbits/s	16-24 Mbits/s	High	3-4
6.4	Component Color Video	Motion Compensation Hybrid	210 Mbits/s	34 or 45 Mbits/s	High	4.5-5 Depending on Motion
6.5	Monochrome 256×256	Scene Decompose	8 bits per pel	≈0.8 bits per pel	High	3-4
6.6	Component Color Video	Conditional Replenishment DPCM	17.6 Mbits/s	2.0 or 1.5 Mbits/s	Moderate	3-5 Depending on Motion
6.7.1	Black & White Document	CCITT 1 Dimen. Coding	1-bit per pel	Compress. Ratio 5-16	Low	Same as Original
6.7.2	Black & White Document	CCITT 2 Dimen. Coding	1-bit per pel	Compress. Ratio 6-32	Moderate	Same as Original

Fig. 6.0.1 Summary of characteristics and capabilities of the systems described in this chapter. Image quality is according to the five point CCIR scale of Table 4.1.

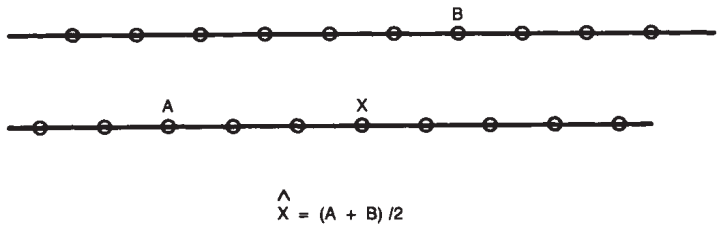


Fig. 6.1.1 Two successive lines of NTSC video signal sampled at three times the color subcarrier frequency of 3579545 Hz. Pel X occurs 3 pels after A and 681 pels after B, and therefore all three pels have the same color subcarrier phase. DPCM uses an average of A and B as $\hat{X}$ the prediction of X.

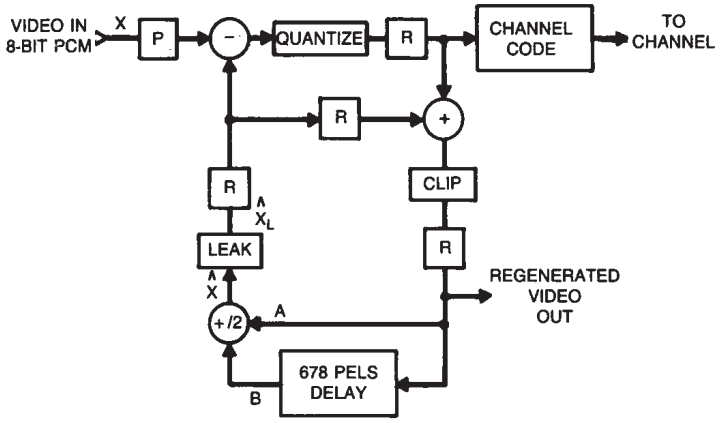


Fig. 6.1.2 Circuit configuration for two-dimensional DPCM Coding. *R* denotes a register. Quantization and Channel coding can be realized via simple Read Only Memories (ROM's). The leak factor is 31/32.

Range of Differential Magnitudes $\lvert X - \hat{X}_L \rvert$	Quantized Values
0-2	±1
3-5	±3
6-10	±7
11-18	±14
19-28	±23
29-40	±34
41-55	±47
56-255	±63

Fig. 6.1.3 Quantization table for prediction errors.

6.1 45 Mbs Intrafield DPCM Codec for NTSC Color Video

The system described here[6.1,6.2] is a fairly simple color TV codec for situations that require a subjectively pleasing picture at the receiver, but which do not need the so called "transparent" coding quality typical of TV studio or network transmission applications. The output bit-rate for video and auxiliary data closely matches the DS3 rate (44.736 Megabits/sec) used in North America. It may also be suitable for some local-area fiberguide or cable TV networks.

The first step is to PCM code the NTSC composite video signal at a sampling rate of three times the color subcarrier frequency using 8-bits per pel. The pel configuration of two successive scan lines is shown in Fig. 6.1.1, and pels A, B and X all have the same color subcarrier phase, i.e., they are equal in regions of constant color. Thus, for DPCM a reasonably good prediction of pel X is the average of pels A and B.

Fig. 6.1.2 shows a block diagram of the DPCM coder. The ROM quantizer outputs one of a limited set of values, 16 in this case, as shown in Fig. 6.1.3. The ROM channel coder then outputs 4-bits per pel to give a video bit-rate of about 43 Megabits/sec, which allows a considerable margin for audio, additional data and stuffing bits to be added prior to the DS3 channel. Fig. 6.1.4 shows the corresponding decoder, for which only a channel decoder ROM is needed.

As was described in Sec. 5.2.3, resilience to transmission bit errors is improved considerably by the use of *leak*, i.e., attenuating the prediction $\hat{X}$ by a small amount before computing the prediction error. However, this can have a detrimental effect on the quality of prediction if the leak is too large (see Sec. 5.2.3k). As a compromise, the leak factor in this system was chosen as 31/32, and was implemented to cause the prediction to decay toward the center of the dynamic range, i.e.,

$$\hat{X}_L = 128 + \frac{31}{32}(\hat{X} - 128) \tag{6.1.1}$$

Note that this is equivalent to subtracting 128 from the input pels and using a standard leak configuration. This arrangement gives virtually no perceptible image degradation for single-bit errors up to an error rate of 10^{-6}.

A clipping circuit at both the coder and decoder prevents overflow and underflow that can be caused by the quantized differential values. For overflow the output is 255; for underflow it is 0. The clipping function also performs another very important service at the decoder during startup and following catastrophic transmission errors. Because of the clipping, discrepancies between the coder and decoder prediction values tend to disappear quite rapidly (within a few line periods), assuming the video signal has a reasonable magnitude.

This system has been implemented in VLSI. The quantizer, channel coder and channel decoder functions were obtained using standard ROM. The remaining operations were designed into a single custom CMOS VLSI using 2.5 micron design rules.

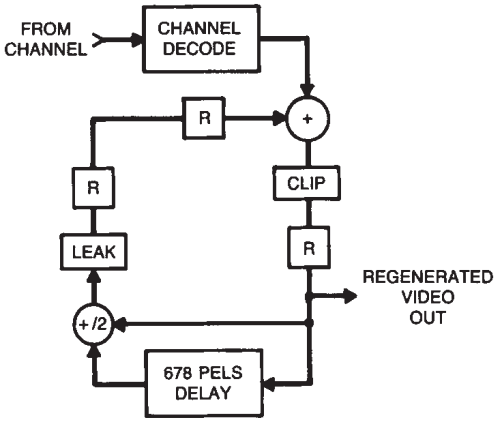
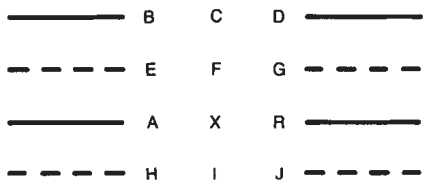


Fig. 6.1.4 Two dimensional DPCM decoder.



$$\hat{X}_1 = (A + D)/2$$

$$\hat{X}_2 = (F + I)/2$$

Fig. 6.2.1 Pel configuration for adaptive DPCM coding. $\tilde{X}_1$ is an intrafield predictor suitable for moving areas, whereas $\tilde{X}_2$ is an interfield predictor suitable for non-moving areas of the picture.

6.2 Adaptive Predictive Interfield Coder for NTSC Color TV

Conceptually, the idea of adaptive prediction is simple, as we saw in Chapter 5. Within each small region of the image the coder must keep track of the directions of maximum correlation between the pels, and then adapt the predictor accordingly. However, for sequentially scanned single images, simple adaptive prediction does not give a marked reduction in bit-rate. The problem is that in low-detail areas of the picture the correlation is fairly high in all directions, and adaptation is of little use, whereas in high-detail areas of the picture the correlation directions change so fast that either the adaptation algorithm cannot keep up or excessive overhead information must be transmitted. The net result is that there has been very little improvement over Graham's original suggestion[5.2.7] of keeping a running track of which is larger, horizontal correlation or vertical correlation, and accordingly using the pel to the left or the pel directly above as a prediction (see Sec. 5.2.3d).

With a 2:1 interlaced television signal however, adaptive predictive coding has a much more marked effect than with sequentially scanned pictures. For example, Fig. 6.2.1 shows a pel configuration along with an intrafield predictor $\hat{X}_1 = (A+D)/2$ and an interfield predictor $\hat{X}_2 = (F+I)/2$. For areas of the picture in which there is no movement $\hat{X}_2$ is the better predictor, whereas in areas of the picture containing significant movement $\hat{X}_1$ is better. This is due largely to the fact that pel X is spatially closer to pels F and I, but temporally closer to pels A and D. An efficient adaptation algorithm simply examines which predictor gave a better prediction of (quantized) pel A and then use that predictor for pel X. This works well because moving areas are usually contiguous in an image.

Such a coder[6.3] is shown in Fig. 6.2.2a. From $\tilde{A}$, the quantized version of pel A, and the two predictions $\hat{A}_1$ and $\hat{A}_2$, which were computed for pel A, a decision is made as to whether $\hat{X}_1$ or $\hat{X}_2$ is the best prediction for pel X. The identical decision can also be made at the receiver with no additional transmitted information as shown in Fig. 6.2.2b. The prediction $\hat{X}$ is then used in the normal manner for DPCM coders.

The use of predictive coding with composite color television signals must take into account the presence of the sinusoidal color subcarrier in the video signal as we saw in Chapter 5. Moreover, the relationship between the pel X (which is to be predicted) and the surrounding previously transmitted pels depends on the sampling pattern of the initial digitization of the composite signal. For example, if the initial sampling rate is exactly four times the NTSC color subcarrier frequency, i.e., ≈14.3 MHz, then in Fig. 6.2.3 pels K, L, M and I are approximately equal to pel X in non-moving regions of constant color. However, this does not preclude the use of other pels in a linear prediction as we saw in Section 5.2.3c.

The simplest intrafield *in-phase* predictor is given by

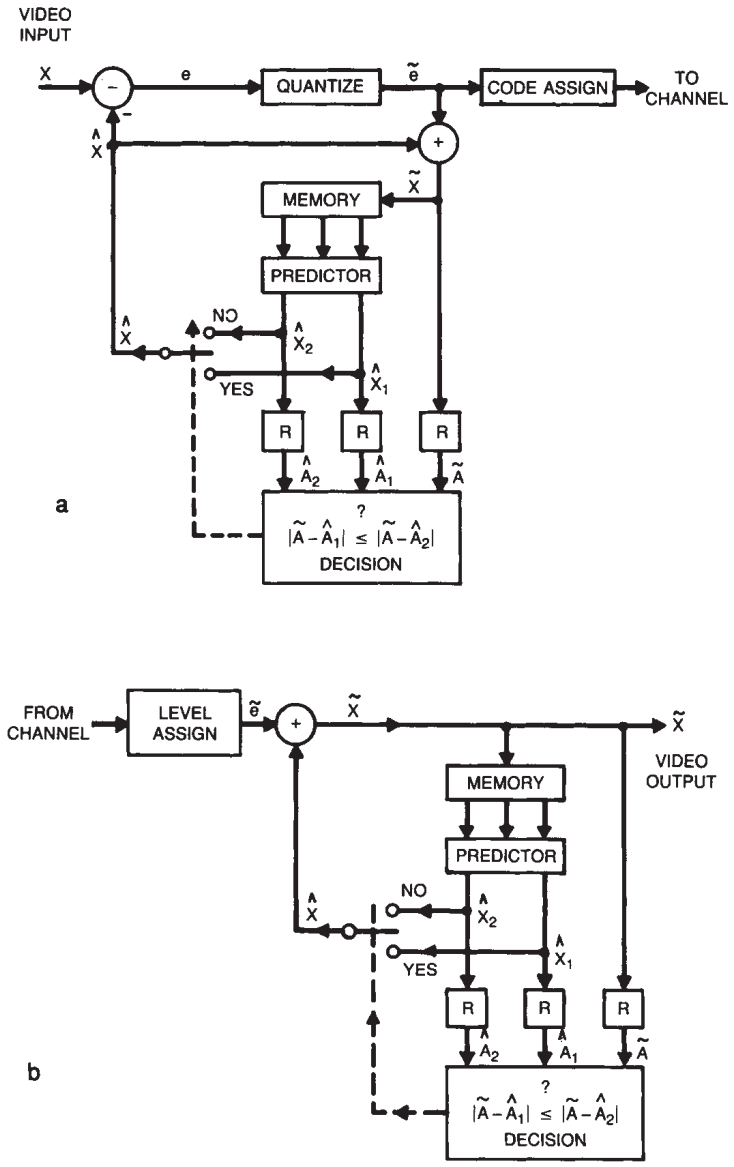


Fig. 6.2.2 Adaptive predictive codec. (a) Coder: Registers *R* are 1 pel delays. (b) Decoder.

$$\hat{X}_1 = M \tag{6.2.1}$$

where pel M of Fig. 6.2.3 is the fourth previous pel along the scan line. This prediction should work well in moving areas of the picture where there is a reduction in spatial resolution.

The suitable interfield in-phase predictor is

$$\hat{X}_2 = \frac{1}{2} I + \frac{1}{4} (K+L) \tag{6.2.2}$$

where pels K, L, and I are in the adjacent lines of the previous field as shown in Fig. 6.2.3. This predictor should work well in stationary areas of the image.

Choosing the best predictor for pel X could be done as shown in Fig. 6.2.2, i.e., choose whichever predictor was best for pel A. However, it has been found that sometimes pel A by itself is not sufficient to enable a good choice of predictors and that additional information should also be taken into account such as which prediction was best for pel C of Fig. 6.2.3. For example, if C_1 was a better prediction of pel C and $\hat{A}_1 \approx \hat{A}_2$, then $\hat{X}_1$ would be chosen as the best predictor for pel X.

In coding a composite color signal with broadcast quality, the usual distortion criterion is that there be no visible quantization noise. Thus, in this system the first step is to PCM code the composite signal at a sampling rate of four times color subcarrier frequency using 9-bits per pel. In addition, the DPCM quantization error must at all times be below the threshold of visibility (see Fig. 5.2.33). Two quantizers were designed for the 10-bit prediction error according to this experimental procedure, and their characteristics are given in Fig. 6.2.4. One is slightly coarser than the other and is used during periods of moderate motion as described below. Both produce subjectively invisible quantization noise. However, measured signal-to-noise is smaller with the coarse quantizer.

If the codec of Fig. 6.2.2 is implemented with variable word-length coding and buffers at the coder and decoder, a buffer overflow strategy must be devised to deal with scenes containing rapid motion. A very straightforward technique is subsampling, e.g., sending only 75 percent or 50 percent of the samples, in order to reduce the data rate until buffer overflow no longer threatens. However, with a composite signal, interpolation is not as simple as with color components or monochrome signals. For example, with 4:3 subsampling, pel P in Fig. 6.2.3 could be interpolated from its (quantized) neighbors by

$$\tilde{P} = \frac{1}{2}(\tilde{N}+\tilde{A}) + \frac{1}{2}(\tilde{N}-\tilde{M}) + \frac{1}{2}(\tilde{A}-\tilde{X}) \tag{6.2.3}$$

$$= \tilde{N} + \tilde{A} - \frac{1}{2}\tilde{M} - \frac{1}{2}\tilde{X}$$

With 2:1 subsampling, better picture quality results if the sampling pattern is shifted between fields, and therefore a suitable interpolation for pel P would be

Fig. 6.2.3 Pel configuration for NTSC composite color TV signal sampled at four times color subcarrier frequency. Pels *K*, *L*, *M*, *I* and *U* have the same color subcarrier phase as pel *X*.

Fine Quantizer		Coarse Quantizer	
Input Magnitude	Quantized Values	Input Magnitude	Quantized Values
0	0	0-1	0
1-2	±1	2-4	±3
3-5	±4	5-7	±6
6-9	±7	8-11	±9
10-14	±12	12-16	±14
15-19	±17	17-21	±19
20-25	±22	22-27	±24
26-32	±29	28-34	±31
33-39	±36	35-42	±38
40-49	±45	43-51	±47
50-59	±54	52-61	±56
60-71	±65	62-73	±67
72-85	±78	74-87	±80
86-101	±93	88-103	±95
102-119	±110	104-121	±112
120-139	±129	122-141	±131
140-162	±150	142-164	±152
163-188	±175	165-194	±177
189-217	±202	195-222	±206
218-251	±233	223-257	±239
252-290	±270	258-297	±276
291-511	±311	298-511	±319

Fig. 6.2.4 Quantization (symmetric) characteristic for 10-bit prediction error input signal. The coarse quantizer is used during periods of moderate motion.

$$\tilde{P} = \frac{1}{2}(\tilde{N}+\tilde{A}) + \frac{1}{4}(\tilde{W}-\tilde{U}) + \frac{1}{4}(\tilde{W}-\tilde{I}) \qquad (6.2.4)$$

$$= \frac{1}{2}(\tilde{N}+\tilde{A}) + \frac{1}{4}(2\tilde{W}-\tilde{U}-\tilde{I})$$

The first term is a luminance estimate obtained from the two pels that are spatially and temporally closest. The second term is a chrominance estimate obtained from the line below in the previous field.

The effects of 4:3 subsampling on picture quality are only visible in areas of high detail where large pel-to-pel luminance transitions occur. Such conditions are usually present only in electronically generated graphics of test waveforms and are extremely rare in moving areas of the picture, which is where subsampling would normally be employed in order to reduce buffer filling.

The effects of 2:1 subsampling are visible at large pel-to-pel luminance transitions and at large line-to-line chrominance transitions. With moderate to rapid motion, which is where 2:1 subsampling would normally be employed, such conditions are extremely rare. The overall codec is shown in Fig. 6.2.5.

Five modes of operation, shown in Fig. 6.2.6, are utilized by the codec in order to keep the buffer from emptying or overflowing. Mode 0 is used exclusively to prevent buffer underflow. With some modification it could also help to alleviate the effects of transmission bit errors. Modes 1 to 3 utilize variable word-length codes to lower the data rate for the quantized prediction errors. For best performance, a slightly different code should be used for the two quantizers so that the data rate is significantly reduced when the coarse quantizer is utilized. Mode 4 has such a low bit-rate that variable word-length coding is not needed.

The buffer size in this system should be large enough to smooth the data over at least one field interval, i.e., 17 msec. At a bit-rate of 42 Mbits/sec, this implies a buffer size greater than 714 kbits. At the receiver, a larger size is required in order to deal with transmission bit errors.

Fig. 6.2.7 shows a mode control strategy based on the fullness of an 800 kbit buffer. Switching to a higher mode occurs when the buffer fullness exceeds the prescribed level, but switching to a lower mode does not take place until the buffer fullness drops below some lower level, as shown. This prevents oscillation between modes, which is detrimental to image quality in some cases.

Operating at a channel bit-rate of about 43 Megabits/sec (3 bits/pel), very high quality images can be transmitted that are free of any visible coding distortion. For normal entertainment television pictures, the codec spends a large majority of its time (more than 80 percent) in mode 1 and most of the remaining time (about 12 percent) in mode 2. Mode 0 is used primarily for low detail pictures, and mode 3 is used only for rapidly moving regions. At a bit rate of 43 Mbits/sec, mode 4 is needed only for anomalous images such as rapidly changing electronically generated signals, and in this case the slightly larger

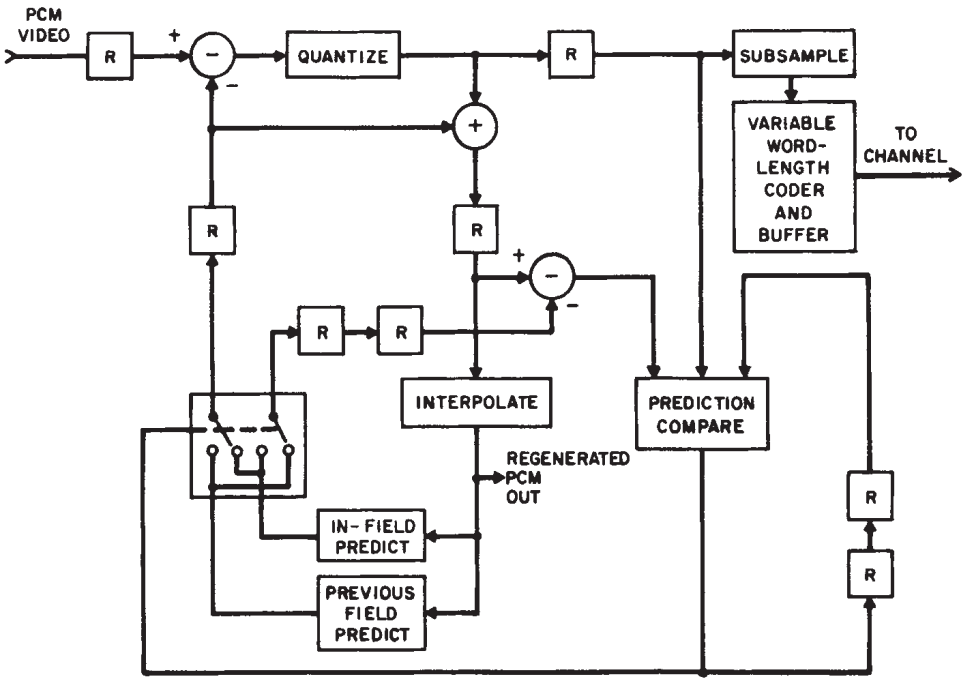


Fig. 6.2.5 Adaptive DPCM coder for NTSC composite color TV signal. The mode controller varies the coarseness of the quantizer and invokes subsampling in order to prevent buffer overflow.

Mode	Code Word Length	Subsample	Quantizer
0	6-bits	none	fine
1	variable	none	fine
2	variable	none	coarse
3	variable	4:3	fine
4	6-bits	2:1	fine

Fig. 6.2.6 Modes used for the codec operating at 42 Mbits/sec.

coding distortion is completely masked by the picture and is invisible to the viewer.

Some additional measures must be taken if there are transmission bit errors. Forward error correction (FEC) channel codecs are the most straightforward solution, and these help to eliminate audio as well as video distortions. However, even with the FEC, occasional catastrophic errors will occur. Leak may be less effective with adaptive coding. In this case, error detection at the receiver and transmission of some PCM data by the transmitter would normally be required in a practical system. More will be said later in this chapter about error control.

Further improvement of adaptive predictive coding for broadcast quality video is certainly possible with more elaborate processing. For example, motion compensation methods are highly desirable for the slow panning and zooming found so often in entertainment television. However, applying motion compensation to a composite color video signal has some fundamental problems, the most important of which is that corresponding translated pels in successive frames do not usually have the same color subcarrier phase (see Sec. 5.2.3j). The usual cure is to either implicitly or explicitly demodulate the composite signal into its color components, and then apply motion compensation to the components[6.4] before reconstructing the composite signal. Unfortunately, this extra composite demodulation almost always introduces some visible artifacts into the picture. Thus, further significant bit-rate reduction for broadcast quality images may have to wait for the conversion of studio equipment to handle digital component signals. Then, codecs will not have to process composite waveforms at all.

6.3 Interfield Hybrid Transform Coder for Component Color TV

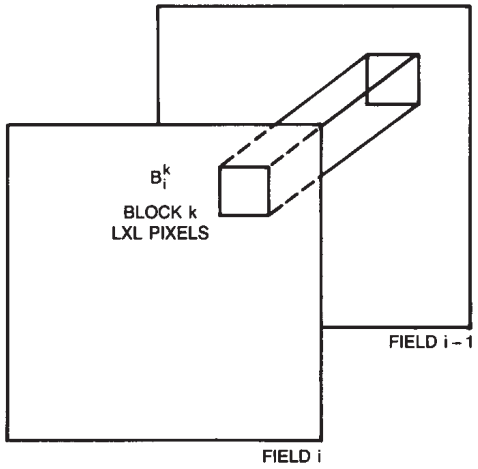
In this section, we discuss an interfield coder design for television (similar to [6.5]) that utilizes hybrid transform coding techniques. Resulting bit-rates are in the range of 10 to 20 Mbits/sec with image quality that is very good, but not transparent as required for network broadcasts. This bit-rate is somewhat below that which can normally be achieved with simple DPCM.

Here we assume that the luminance signal Y as well as two chrominance signals, e.g., I and Q, are available to the coder. This would be the case, for example, in a videoconferencing situation where the camera and coder were in close proximity, or alternatively in a networking application where the studio produced digital component signals. Thus, artifacts due to modulation and demodulation of a color composite signal need not be taken into account in the coding algorithm.

However, interlace does affect both the performance and the design of the coder. We could, in principle, convert each TV frame from a 2:1 interlaced format to a sequentially scanned, non-interlaced format, and then employ intraframe transform or intraframe hybrid transform coding. However, such an

Switch Mode From	To	Buffer Content (kbits)
0	1	>102
1	2	>614
2	3	>691
3	4	>768
4	3	<717
3	2	<640
2	1	<563
1	0	<77

Fig. 6.2.7 Mode control parameters for a buffer size of 800 kbits.



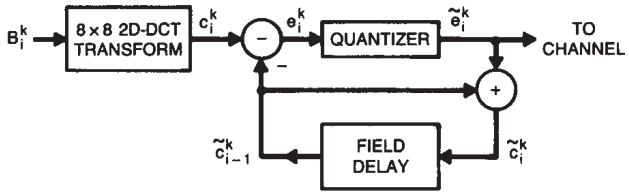


Fig. 6.3.1 Field differential hybrid transform coder for component color TV. Blocks of 8×8 pels are transformed via the 2D-DCT. The coefficients are then coded using field differential PCM.

approach would add another level of complexity without much promise of improved performance, especially in light of the fact that movement in the scene would markedly reduce line-to-line correlation upon which intraframe coders depend so heavily.

Thus, the basic approach of this coder (shown in Fig. 6.3.1) is to partition each field into two-dimensional blocks, transform each block using the separable discrete-cosine-transform (DCT) and then code each transform coefficient using simple field-differential DPCM, i.e., predictive coding using the corresponding coefficient of the previous field as a prediction. In the following, we will dwell mostly on the luminance, since most of the coding bits will be devoted to that component. Chrominance coding will be treated briefly thereafter.

The choice of block size is affected by several factors. A large size leads to a smaller bit-rate in low-detail areas of the picture. However, it also means higher complexity and less ability to adapt to changing picture conditions (see Sec. 5.3.1d). A block size of 8×8 pels has been found to be a reasonable compromise, and this is shown in Fig. 6.3.1.

With interlace, horizontal correlation should exceed vertical correlation within a transform block. However, line-to-line spacing within a *frame* is half the line spacing within a field. Thus, correlation between blocks in successive fields will exceed the pel-to-pel correlation within a block, unless there is rapid horizontal movement in the scene, in which case intra-block pel-to-pel correlation will be increased significantly because of camera integration, which causes blurring of moving areas.

The NTSC luminance signal,* sampled at ≈8.4 MHz produces about 534 pels per scan line. However, nearly 18 percent of these are in the horizontal blanking period, and therefore only about 440 pels per scan line need be coded and transmitted. Similarly, of the 262.5 lines per field, 5 to 8 percent of them are in the blanking period, and therefore only 248 lines/field need be sent. Thus, each field consists of a 31×55 array of blocks.

Improvement in performance with this coder would require at least adaptive quantization, variable word-length (entropy) coding and very large buffer memories at the transmitter and receiver, all of which lead to considerable expense and system complexity. Some compromises are clearly in order.

We saw in Sections 5.3-4 that if the mean square values $\{ \sigma_m^2 \}$ are known for the N quantities $\{ c_m \}$ that are to be coded and transmitted, then the mean square error could be minimized by sending some fraction $p < 1$ of the coefficients and coding each of them at an average rate of r_m bits per coefficient. In particular, using uniform quantization and entropy coding (MSV's arranged

---

* Line scan rate ≈15734 Hz, bandwidth ≈4.2 MHz, 525 lines per frame, image aspect ratio 4:3.

Luminance DCT Coefficient MSV's (16 pix average)

1.9e+05	1.1e+04	2.5e+03	1.3e+03	5.4e+02	2.2e+02	1.3e+02	3.9e+01
1.2e+04	1.3e+03	3.7e+02	1.4e+02	5.8e+01	2.3e+01	9.0e+00	4.9e+00
4.6e+03	5.5e+02	2.2e+02	9.3e+01	4.6e+01	2.4e+01	9.2e+00	1.3e+01
1.5e+03	2.6e+02	1.2e+02	5.8e+01	3.0e+01	1.5e+01	6.4e+00	6.0e+00
5.4e+02	1.4e+02	7.7e+01	4.3e+01	2.5e+01	1.5e+01	6.2e+00	8.0e+00
5.1e+02	8.2e+01	5.3e+01	3.1e+01	2.1e+01	1.4e+01	5.9e+00	9.8e+00
4.9e+02	5.1e+01	3.1e+01	2.0e+01	1.2e+01	7.2e+00	3.5e+00	3.4e+00
2.5e+02	3.6e+01	2.9e+01	2.0e+01	1.6e+01	1.3e+01	5.5e+00	1.2e+01

Corresponding Field Differential MSV's

6.7e+02	5.8e+01	2.7e+01	1.6e+01	1.2e+01	1.2e+01	5.1e+00	1.2e+01
8.3e+02	8.5e+01	3.6e+01	1.7e+01	9.3e+00	5.4e+00	3.1e+00	2.7e+00
4.3e+02	9.9e+01	5.4e+01	3.2e+01	2.3e+01	2.1e+01	8.2e+00	2.1e+01
3.1e+02	9.2e+01	5.0e+01	2.9e+01	1.8e+01	1.2e+01	5.8e+00	9.3e+00
5.5e+02	8.8e+01	5.6e+01	3.4e+01	2.4e+01	1.7e+01	7.4e+00	1.4e+01
8.1e+02	8.6e+01	6.1e+01	3.7e+01	2.8e+01	2.2e+01	9.2e+00	1.8e+01
6.8e+02	6.6e+01	4.5e+01	2.9e+01	1.9e+01	1.2e+01	6.0e+00	6.2e+00
3.6e+02	6.2e+01	5.4e+01	3.7e+01	2.9e+01	2.6e+01	1.0e+01	2.5e+01

Fig. 6.3.2 Mean Square Values (MSV's) of luminance DCT coefficients and their corresponding field differentials averaged over 16 images containing no movement.

5 4 3 2 1 0 1 1	4 2 1 0 0 0 0 0	3 1 0 0 0 0 0 0
4 3 3 2 1 0 1 1	3 1 0 0 0 0 0 0	2 1 0 0 0 0 0 0
4 3 2 1 1 0 1 1	2 1 0 0 0 0 0 0	2 0 0 0 0 0 0 0
4 3 2 1 1 0 1 1	2 1 0 0 0 0 0 0	1 0 0 0 0 0 0 0
4 2 2 1 1 0 0 1	2 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
4 2 2 1 0 0 0 0	1 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0
3 2 2 1 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
3 2 2 1 0 0 1 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
Y	*I*	*Q*

Fig. 6.3.3 Example of nonadaptive hybrid field differential quantizer bit allocation for the *Y*, *I*, and *Q* color components. Average bit-rate is 2 bits/pel.

BITS/PEL	PSNR = $20\log\left(\frac{255}{\sqrt{MSE}}\right)$
0.5	29 dB
1.0	32 dB
2.0	38.5 dB
3.0	42.6 dB

Fig. 6.3.4 Performance of nonadaptive hybrid field differential coding. Peak signal to rms noise ratio (unweighted PSNR) is shown versus bit-rate for a scene containing fairly rapid motion.

Luminance DCT Coefficient MSV's (16 pix's containing movement)

7.3e+04	5.0e+03	1.8e+03	8.6e+02	6.0e+02	3.4e+02	5.1e+02	1.2e+02
1.1e+04	8.9e+02	3.3e+02	1.9e+02	1.2e+02	6.3e+01	4.9e+01	2.9e+01
4.0e+03	3.7e+02	1.8e+02	9.4e+01	5.3e+01	3.3e+01	2.4e+01	1.5e+01
1.8e+03	3.0e+02	1.6e+02	7.7e+01	4.6e+01	3.0e+01	2.4e+01	1.6e+01
1.3e+03	1.9e+02	1.1e+02	5.8e+01	4.0e+01	2.7e+01	2.0e+01	1.4e+01
5.2e+02	1.1e+02	6.3e+01	4.5e+01	3.3e+01	2.0e+01	1.5e+01	9.5e+00
1.0e+03	1.3e+02	7.3e+01	4.9e+01	3.5e+01	2.1e+01	1.8e+01	1.3e+01
8.3e+02	1.1e+02	6.0e+01	3.7e+01	2.6e+01	1.8e+01	1.6e+01	1.4e+01

Corresponding Field Differential MSV's

2.0e+03	6.8e+02	3.7e+02	1.8e+02	1.1e+02	6.8e+01	6.3e+01	4.6e+01
1.1e+03	3.6e+02	2.2e+02	1.2e+02	6.4e+01	3.8e+01	3.4e+01	2.7e+01
6.6e+02	2.5e+02	1.5e+02	8.9e+01	4.8e+01	3.0e+01	3.0e+01	2.3e+01
1.6e+03	3.1e+02	1.9e+02	9.9e+01	6.0e+01	4.2e+01	3.7e+01	2.9e+01
1.1e+03	2.2e+02	1.4e+02	8.0e+01	5.4e+01	3.8e+01	3.4e+01	2.2e+01
4.0e+02	1.4e+02	9.5e+01	7.0e+01	5.3e+01	3.7e+01	2.9e+01	1.7e+01
1.4e+03	2.3e+02	1.3e+02	8.1e+01	6.0e+01	4.1e+01	3.4e+01	2.5e+01
1.2e+03	2.1e+02	1.1e+02	7.2e+01	5.1e+01	3.4e+01	3.0e+01	2.5e+01

Fig. 6.3.5 Mean Square Values (MSV's) of luminance DCT coefficients and field differentials averaged over 16 frames of a scene containing motion.

Luminance DCT Coefficient MSV's (16 pix's moving area only)

6.6e+04	6.2e+03	1.7e+03	6.4e+02	3.5e+02	2.9e+02	2.6e+02	5.7e+01
2.1e+04	1.6e+03	5.4e+02	2.4e+02	1.4e+02	7.9e+01	6.4e+01	3.1e+01
1.1e+04	7.5e+02	2.5e+02	1.1e+02	5.1e+01	3.6e+01	2.9e+01	1.6e+01
5.4e+03	5.6e+02	2.6e+02	1.3e+02	5.2e+01	5.5e+01	4.0e+01	2.4e+01
3.2e+03	3.7e+02	1.5e+02	8.0e+01	3.5e+01	2.9e+01	2.3e+01	1.5e+01
1.4e+03	1.5e+02	8.6e+01	3.7e+01	1.6e+01	1.2e+01	1.0e+01	6.0e+00
4.4e+03	2.7e+02	1.0e+02	7.5e+01	3.7e+01	3.8e+01	2.9e+01	1.8e+01
3.2e+03	2.6e+02	1.0e+02	5.7e+01	3.0e+01	3.1e+01	2.8e+01	1.5e+01

Corresponding Field Differential MSV's

1.0e+04	2.2e+03	1.1e+03	4.5e+02	2.2e+02	1.1e+02	1.5e+02	1.1e+02
4.0e+03	9.9e+02	6.3e+02	2.6e+02	1.1e+02	5.1e+01	5.0e+01	4.1e+01
2.0e+03	6.9e+02	3.9e+02	1.8e+02	7.1e+01	5.2e+01	3.9e+01	3.5e+01
7.0e+03	1.0e+03	6.0e+02	2.4e+02	1.2e+02	9.2e+01	6.7e+01	6.1e+01
4.5e+03	6.1e+02	3.6e+02	1.5e+02	7.4e+01	4.6e+01	4.1e+01	3.6e+01
8.8e+02	2.1e+02	1.3e+02	8.7e+01	3.7e+01	3.0e+01	1.9e+01	1.7e+01
6.3e+03	7.1e+02	3.7e+02	1.7e+02	1.0e+02	6.6e+01	5.3e+01	5.2e+01
5.8e+03	6.7e+02	3.4e+02	1.5e+02	8.9e+01	6.2e+01	5.4e+01	4.5e+01

Fig. 6.3.6 MSV's of luminance DCT coefficients and field differentials averaged over the moving areas of 16 frames containing movement.

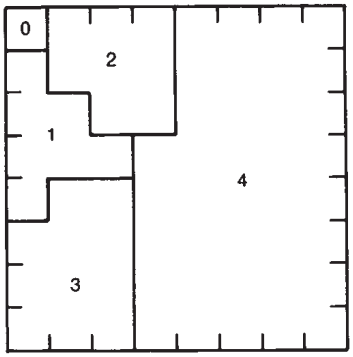


Fig. 6.3.7 Each block of 8×8 DCT coefficients is partitioned into five sub-blocks, and the field differential MSV's are measured within each sub-block to enable adaptive quantization depending on temporal and spatial activity. Sub-blocks are not used for chrominance.

in decreasing order)

$$r_m \approx \frac{1}{1.39} \ln \frac{\sigma_m^2}{\sigma_{pN}^2} \text{ bits.} \tag{6.3.1}$$

The quantizer step size for all coefficients is given by

$$\Delta \approx 3.81 \ \sigma_{pN} \tag{6.3.2}$$

Similar results were obtained for the Lloyd-Max quantizer.

As an example, consider Fig. 6.3.2, which shows DCT luminance coefficient MSV's plus the corresponding hybrid field differential MSV's averaged over 16 monochrome pictures containing no movement. As expected, the vertical spatial frequencies show higher energies than the horizontal because of the larger spacing between lines of a field. Also as expected, the total energy in the hybrid field differentials is significantly less than that of the DCT coefficients themselves. However, we also see that many of the hybrid field differential MSV's (particularly at the higher vertical frequencies) are larger than their DCT counterparts, which indicates that there is not much field-to-field correlation for those particular coefficients.

If Lloyd-Max quantization is used, the number of bits per coefficient is constant. Fig. 6.3.3 shows an example of hybrid field-differential quantizer bit allocation for the Y, I, and Q signals. In this case the average bit-rate is two bits/pel. Fig. 6.3.4 shows the luminance PSNR (unweighted) at various bit-rates using Lloyd-Max quantizers to code a scene containing fairly rapid motion.[6.5] We see that for reasonably good picture quality (35-40 dB PSNR) close to two bits/pel or 13 Mbs are required. For very good picture quality (40-45 dB PSNR) about 3 bits/pel or 20 Mbs are needed for luminance.

Fig. 6.3.5 shows DCT luminance coefficient MSV's along with hybrid field differential MSV's averaged over 16 frames containing two people in motion. Although the DCT coefficient MSV's are comparable to those of Fig. 6.3.2, we see that the hybrid field differential MSV's are somewhat larger. In fact, if we measure MSV's only over the moving areas of the pictures, as shown in Fig. 6.3.6, we see that the moving-area hybrid field differentials are, for the most part, considerably larger than stationary-area field differentials of Fig. 6.3.2. Moreover, most of the moving-area field differentials are larger than their corresponding DCT coefficients, although the total moving-area field differential energy is still less than the coefficient energy.

Therefore, not only do we have to code the transmitted values differently from each other because of dissimilar MSV's, we should also code the moving-areas of the picture differently than the stationary areas. Moreover, some of the blocks are "low-detail" with low MSV's, while other blocks are "high-detail" having larger MSV's.

One way of adapting to these changing conditions is to partition the block of hybrid differentials to be coded into sub-blocks, and measure the sum of the

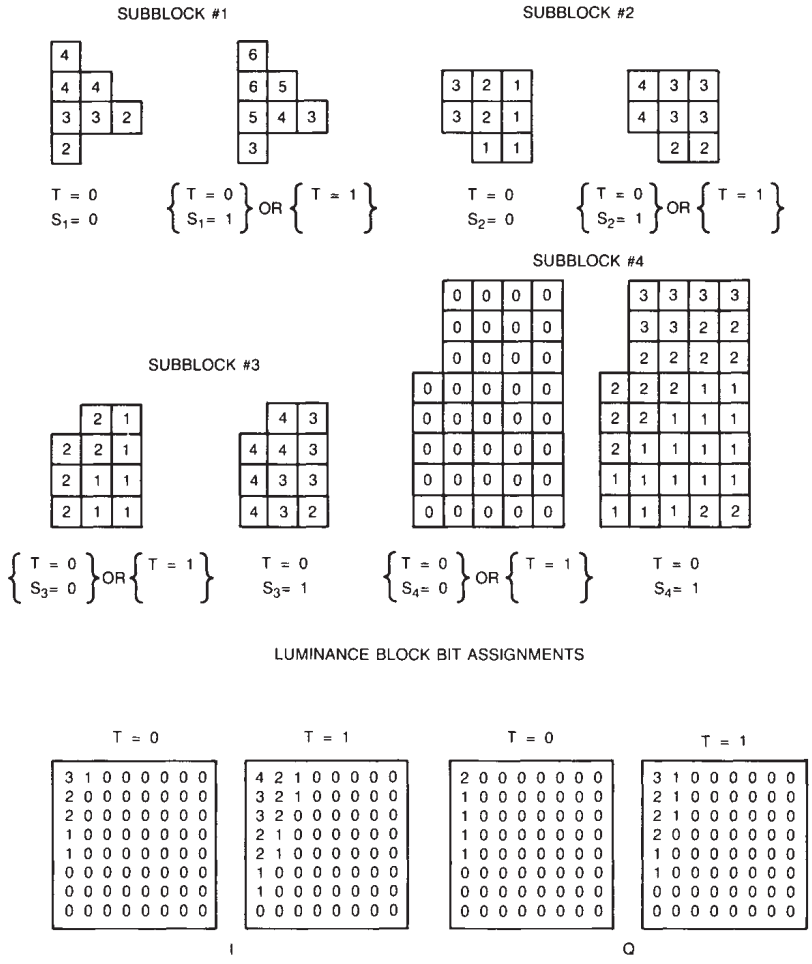


Fig. 6.3.8 Quantizer bit assignments for the sub-blocks of Fig. 6.4.7. For temporally active blocks, sub-blocks 1 and 2 are coded with more bits. Otherwise, coding is according to spatial activity. Sub-blocks are not used for chrominance.

BITS/PEL	PSNR
0.89	33.7 dB
1.37	36.4 dB
1.87	39.5 dB

Fig. 6.3.9 Performance of adaptive hybrid field differential coding. Unweighted PSNR is shown versus bit-rate for the same scene used in Fig. 6.4.4.

squared differential values within each sub-block in order to determine how the coefficients should be coded. For example, Fig. 6.3.7 shows partitioning of an 8×8 block into five sub-blocks.[6.5] If the DC squared field differential value of sub-block 0 exceeds a predefined threshold, then the entire block is considered temporally active, and parameter T is set to 1. Otherwise, $T = 0$. If the sum of the squared field differential values of sub-block i exceeds a predefined threshold, then that sub-block is considered active and $S_i = 1$. Otherwise, $S_i = 0$. The sub-blocks then have their field differential values quantized and coded according to their activity classification.

Fig. 6.3.8 shows an adaptive Lloyd-Max quantizer bit assignment designed for an overall average bit-rate of 2.0 bits/pel. The DC field differential value (sub-block 0) is always coded with 8-bits. For temporally active blocks ($T = 1$), sub-blocks 1 and 2 are coded using more bits while sub-blocks 3 and 4 are coded using fewer bits. For temporally inactive blocks ($T = 0$), the sub-blocks are coded individually using variable bits depending on whether they are active or non-active, respectively. The chrominance differential values are coded with variable bits depending only on whether the block is temporally active or temporally non-active, respectively.

Fig. 6.3.9 shows luminance PSNR (unweighted) for this adaptive coding algorithm[6.5] operating on the same color TV sequence used in Fig. 6.3.4. Comparison shows an improvement with adaptive coding of only 1 to 2 dB. However, the adaptive algorithm is less sensitive to image statistics and handles scene changes better than the nonadaptive algorithm.

6.4 Adaptive Interframe Transform Coder for Contribution-Quality Applications - CCIR 723

The CCIR* has standardized a component video coder, designated as CCIR 723, for video applications requiring contribution quality, i.e., near transparent reproduction. The source video is CCIR 601 Y Cb Cr components, with pel values biased downward to the range -128 to 127 prior to coding. CCIR 723 is commonly used at bit rates in excess of 30 Mbs for 50 fields/sec and 40 Mbs for 60 fields/sec.

The coding algorithm is a straightforward extension of the previous Section, with several additional features for improving coding efficiency. Most significantly, the prediction is carried out in the pel domain. This requires an additional 8×8 inverse discrete cosine transform (IDCT), as shown in Fig. 6.4.1. However, the ability to easily perform pel-domain motion-compensated prediction more than offsets the additional complexity of the IDCT. CCIR 723

---

* Now known as International Telecommunication Union - Radio Sector, abbreviated ITU-R.

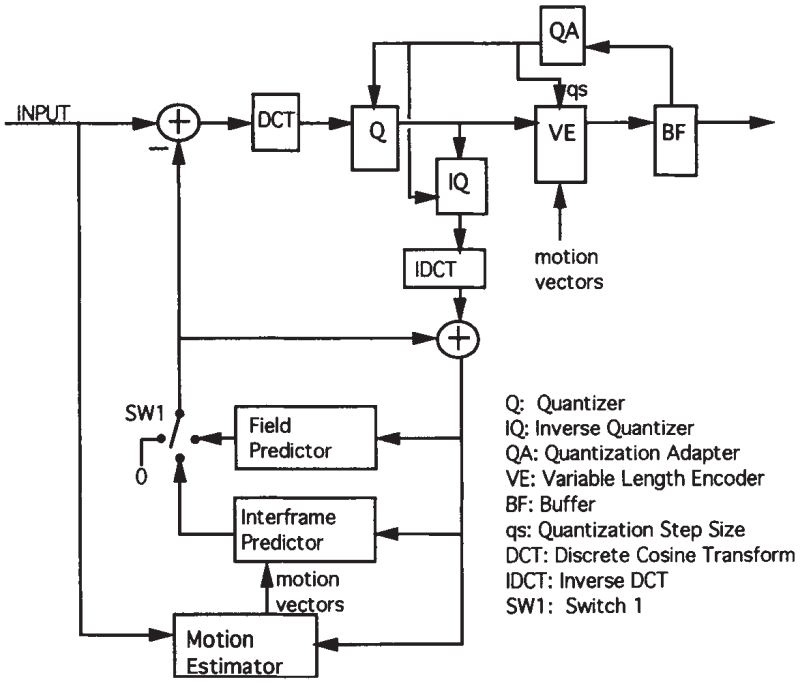


Fig. 6.4.1 Adaptative interframe transform coder CCIR 723. For each Macroblock three coding modes are available - intrafield, interfield and motion compensation using decoded pels from two fields back.

defines a Block as 8×8 samples. It also defines a Macroblock as 16 pels by 8 lines, including luminance and chrominance. Thus, a Macroblock consists of two horizontally adjacent Y-Blocks, plus one Cb-Block and one Cr-Block, i.e., 256 samples in all. A Stripe is a complete horizontal row of 45 Macroblocks. Each field of video has either 288 coded lines (50 Hz) or 248 coded lines (60 Hz). For the latter, the first five lines are not active and are discarded by the decoder.

For each Macroblock, three coding modes are available. The first is simple Intrafield transform coding, where Switch 1 of Fig. 6.4.1 is set to the zero position for the duration of the Macroblock. The second is Interfield prediction, where each pel of the Macroblock is predicted using an average of the pel above and the pel below in the previous decoded field.

The third coding mode incorporates Motion Compensated Interframe prediction using the decoded pels from two fields back in time. One Motion Vector per Macroblock is transmitted and used to predict all pels in the Macroblock. The Motion Vector range is ± 15.5 Y pels horizontally, and ± 7.5 lines vertically, with half pel accuracy. The predictions for noninteger Motion Vectors are computed using bilinear interpolation, i.e., linear interpolation horizontally followed by linear interpolation vertically. Note that for color pels, horizontal interpolation with quarter pel accuracy is needed.

The method used by an encoder for choosing the Macroblock coding mode is not specified by the standard. One possibility is to encode each Macroblock by all three modes and select the one producing the fewest bits. A somewhat simpler method is to compute the mean square (or absolute) prediction error for each mode and select the mode producing the smallest value.

Motion Vectors are DPCM coded (no quantization, however) using the Motion Vector to the left as a prediction. Zero prediction is used for the leftmost Macroblock. The horizontal and vertical components of the differential Motion Vectors are then coded separately using Huffman coding. Codewords are needed for all values between ± 31.0. Codewords range from 2 to 14 bits in length.

Adaptive quantization for CCIR 723 utilizes many of the principles described in previous chapters. First, frequency dependent log Visibility Threshold matrices are defined for luminance and chrominance as shown in Fig. 6.4.2. The large asymmetry of the Y matrix is due to the Y pels being much closer horizontally than vertically. Thus, horizontal luminance DCT spatial frequencies can withstand more quantization noise than equivalent vertical spatial frequencies.

Next, a four level Activity Function is defined that is used to augment the Visibility Thresholds for each Macroblock according to some measure of local masking. The masking measure is outside the standard. In principle, spatial and/or temporal masking could be used to determine the Activity Function.

m \ n	0	1	2	3	4	5	6	7
0	00	00	02	08	12	18	22	28
1	00	06	06	10	16	18	22	34
2	00	06	10	14	18	20	24	38
3	02	06	12	16	18	20	26	40
4	06	12	14	16	20	22	28	42
5	10	14	14	18	22	24	30	42
6	14	16	16	18	22	24	34	44
7	14	18	18	20	24	30	38	44

Initial Luminance Thresholds

m \ n	0	1	2	3	4	5	6	7
0	00	00	03	04	06	08	08	11
1	00	01	02	03	06	08	09	13
2	02	02	03	04	07	09	10	16
3	03	04	05	05	08	10	12	16
4	05	06	06	07	09	11	13	17
5	08	07	09	09	11	14	16	21
6	10	11	11	11	14	16	19	24
7	12	12	12	12	17	18	20	26

Initial Chrominance Thresholds

Fig. 6.4.2 Frequency dependent log Visibility Thresholds for luminance and chrominance DCT coefficients. A higher Threshold implies a lower visibility of quantization noise. These values are augmented according to a local Activity Function to obtain T_{mn}.

Quantizer Index	Quantized Value	Quantizer Index	Quantized Value
0	0	384	513
1	1	385	517
2	2	386	521
.	.	.	.
.	.	.	.
.	.	.	.
255	255	511	1021
256	256	512	1027
257	258	513	1035
258	260	514	1043
.	.	.	.
.	.	.	.
.	.	.	.
383	510	639	2043

Fig. 6.4.3 Companded quantizer characteristic. Uniformly quantized integer values in the range -2048 to 2032 are rounded to the nearest allowed representation level. Only positive levels are shown. The 1279 levels are numbered by a quantization index I_{mn} in the range -639 to +639.

m \ n	0	1	2	3	4	5	6	7
0	00	02	06	12	20	28	36	44
1	01	05	11	19	27	35	43	51
2	03	07	13	21	29	37	45	52
3	04	10	18	26	34	42	50	57
4	08	14	22	30	38	46	53	58
5	09	17	25	33	41	49	56	61
6	15	23	31	39	47	54	59	62
7	16	24	32	40	48	55	60	63

Scanning Order for Luminance

m \ n	0	1	2	3	4	5	6	7
0	00	02	03	09	10	20	21	35
1	01	04	08	11	19	22	34	36
2	05	07	12	18	23	33	37	48
3	06	13	17	24	32	38	47	49
4	14	16	25	31	39	46	50	57
5	15	26	30	40	45	51	56	58
6	27	29	41	44	52	55	59	62
7	28	42	43	53	54	60	61	63

Scanning Order for Chrominance

Fig. 6.4.4 Scanning path for quantized DCT coefficients.

For the lowest Activity level, Y thresholds are clipped to a maximum of 24, and Cb/Cr are clipped to a maximum of 9. For the next Activity level, Y thresholds are clipped to a maximum of 34 and Cb/Cr are clipped to a maximum of 16. For the next Activity level all thresholds are incremented by 2, and for the highest Activity level all thresholds are incremented by 8. The value of the Activity Function is transmitted for each Macroblock so that the decoder is also able to derive the augmented log Visibility Thresholds T_{mn}.

The next control of quantization is effected by a buffer fullness factor F that is transmitted once per stripe. The range of F is 0 to 175.

Given the buffer fullness factor F and the augmented log Visibilities T_{mn}, we first compute an interim log Visibility q_{mn} using an empirically derived relation

$$q_{mn} = Min[2T_{mn} - 48, F] + F \tag{6.4.1}$$

Following this, q_{00} is clipped to be in the range 0 to 48, q_{mn} is clipped to be in the range 0 to 175, and the actual quantization step size for each DCT coefficient is computed by

$$\Delta_{mn} = 2^{[q_{mn}/16 - 1]} \tag{6.4.2}$$

The range of Δ_{mn} is 0.5 to 980. It is computed to an accuracy of 12-bits.

A first mid-step uniform quantization is carried out by simply dividing each DCT coefficient by Δ_{mn} and rounding to the nearest integer. A final companded quantization is then effected by rounding the magnitude to the nearest representation level in Fig. 6.4.3. At the decoder, the quantized DCT coefficients $\tilde{c}_{mn}$ are simply the product of the (signed) representation levels and Δ_{mn}.

Assuming prediction errors are clipped to the range -128 to 127, DCT coefficients have the range -1024 to 1016 before quantization, -2048 to 2032 after uniform quantization, and an allowable range of -2043 to +2043 after companded quantization.

Each of the 64 coefficients in a block is thus represented by a quantizer index I_{mn} in the range -639 to +639. The quantizer indices for each block are transmitted in the order shown in Fig. 6.4.4, which approximates the order of decreasing coefficient variance for luminance and chrominance blocks.

Each nonzero index is assigned its own variable length codeword. However, zero indices are runlength coded with variable length codes assigned to each possible runlength between 1 and 63. The last runlength, if any, of each block is not transmitted. Instead an End-of-Block (EOB) code tells the decoder that all remaining coefficients of the block are zero. An EOB is sent for each block even if there is no last runlength.

If two runlengths are separated by one or more indices of value +1, one of the indices is not transmitted. At the decoder an extra index of value +1 is inserted.

CODEWORD (base 4)	Indices and Runlengths (r) Y	Y&C	C	CODEWORD (base 4)	Indices and Runlengths (r) Y	Y&C	C
1		1		0		-1	
30		2		21		-2	
31		r1		20		r2	
320		r3		231		r4	
321	3		r5	230	-3		r6
330	4		r7	221	-4		r8
331		EOB1		220		EOB0	
3220	r5		3	2331	r6		-3
3221	r7		4	2330	r8		-4
3230		r9		2321		r10	
3231		r11		2320		r12	
3320		5		2231		-5	
3321	6		r13	2230	-6		r14
3330	7		r15	2221	-7		r16
3331	8		r17	2220	-8		r18
32220	r13		6	23331	r14		-6
32221	r15		7	23330	r16		-7
32230	r17		8	23321	r18		-8
32231		r19		23320		r20	
32320		r21		23231		r22	
32321		r23		23230		r24	
32330		r25		23221		r26	
32331		r27		23220		r28	
33220		9		22331		-9	
33221		10		22330		-10	
33230		11		22321		-11	
33231		12		22320		-12	
33320		13		22231		-13	
33321		14		22230		-14	
33330		15		22221		-15	
33331		16		22220		-16	
322220		r29		233330		r30	
322221		r31		233331		r32	
322230		r33		233320		r34	
322231		r35		233321		r36	
322320		r37		233230		r38	
322321		r39		233231		r40	
322330		r41		233220		r42	
322331		r42		233221		r44	
323220		r45		232330		r46	
323221		r47		232331		r48	
323230		r49		232320		r50	
323231		r51		232321		r52	
323320		r53		232230		r54	
323321		r55		232231		r56	
323330		r57		232220		r58	
323331		r59		232221		r60	
332220		r61		223330		r62	
332221		r63		223331		null	

Fig. 6.4.5 A portion of the Variable Length Code for Quantizer Indices and Runlengths. Only the first 98 codewords are shown. Codewords are shown in base 4, i.e., two bits per quarternary symbol.

For the shorter codewords, different variable length codes are used for luminance and chrominance coefficients. Fig. 6.4.5 shows some of these. For index values larger in magnitude than 16, an algorithmic construction of codewords is provided to alleviate the need for large lookup tables.

Two EOB codewords are provided for synchronization purposes. For each block one or the other is chosen based upon a predetermined selection rule known to both encoder and decoder. If the wrong EOB is received, the decoder knows an error occurred.

6.5 Decomposition of Single Images into Edges and Texture

We have seen that under virtually no circumstances can images (of any interest) be modeled as two-dimensional, stationary, Gaussian random fields. Instead they tend to consist of distinct objects, such as persons, lungs or grassy fields, which have fairly well defined and abrupt boundaries and whose interiors are more or less random texture. Unfortunately, we have also seen that rate-distortion theory is most easily applied to stationary, Gaussian random processes. Moreover, transform coding is also most efficient on images that do not contain many large luminance transitions such as occur at object boundaries. One approach to resolving this dilemma is to decompose the visual information into two components, i.e.,

$$B(x,y) = d(x,y) + r(x,y) \tag{6.5.1}$$

where $d(x,y)$ is an image containing basically objects with their interior texture removed, and $r(x,y)$ is a remainder image containing texture, surface roughness and other irregularities of the object interiors.[6.6] Thus, in the $d(x,y)$ image, relatively large luminance transitions occur at the object boundaries, whereas the interiors of objects are relatively smooth and uniformly colored. In the $r(x,y)$ image, the luminance transitions are much smaller and noise-like.

Hopefully, $d(x,y)$ can then be coded efficiently using algorithms such as interpolative coding, whereas $r(x,y)$ will approximate a Gaussian random field that can be efficiently represented using transform coding techniques. We also desire that mean square error (or at least frequency-weighted MSE) be a meaningful distortion criterion for the transform coding. This is facilitated if a compression-type nonlinearity, e.g., log or cube-root, is applied either to $B(x,y)$ or $r(x,y)$ prior to coding so that at least Weber's Law effects are compensated.

Construction of the object boundary function $d(x,y)$ is not straightforward for several reasons. For example, at some points on the object boundary, the object intensity may be nearly equal to that of the background scene behind the object, which can lead to an incorrect object boundary definition. Also, distinguishing between a distinct small object and a piece of texture within a large object, is a design decision that depends on the final desired picture quality as well as the relative proportion of bits devoted to $d(x,y)$ and $r(x,y)$. If the smallest allowable object size is too small, then $d(x,y)$ will require too many

bits, whereas if only large objects are allowed, then $r(x,y)$ will require too many bits.

Thus, the edge detection algorithm implicit in the construction of $d(x,y)$ must include a smoothing operation as well as spatial differentiation. Many edge detection schemes are contained in the pattern recognition literature. An example is shown in Fig. 6.5.1. First, a spatial differentiation is carried out on the image.* In this case, each pel is replaced by the difference between itself and an average of its four nearest neighbors. The magnitudes of the differential values are then thresholded with the result shown in Fig. 6.5.1b, i.e., pels for which the absolute value of the differential exceeds some predetermined threshold are shown in black. A smoothing operation is then carried out in order to eliminate small objects and unclosed contours. The result defines the object boundaries, and is shown in Fig. 6.5.1c.

The next step in constructing $d(x,y)$ is to approximate the image intensity on even numbered scan lines (exemplified in Fig. 6.5.2) by a piecewise linear function whose breakpoints are at or near the boundary pels defined previously. Note that in this case the *boundary* may consist of several pels, and that a linear approximation of the boundary pels is calculated as well.

For each odd numbered scan line, an interpolated piecewise linear approximation is formed from those of the adjacent even lines above or below. A check is then carried out to see how close this interpolated approximation is to the original intensity. If it is not good enough (according to some predefined criterion), then a separate piecewise linear approximation is formed for that odd line in the same way as was done for even lines. The resulting piecewise linear approximations on each scan line then define the *object* image intensity $d(x,y)$.

The breakpoints of $d(x,y)$ may be coded using any of the two-level graphics coding methods described in earlier chapters (see Sec. 5.7). For example, in [6.6] a two-dimensional, run-length, Huffman coding scheme is utilized. The amplitudes of $d(x,y)$, which are usually quantized more coarsely than the original image, must be sent separately.

Coding the remainder function $r(x,y)$ can be accomplished with good efficiency by the use of transform coding. For a given picture quality, the techniques described in Chapter 5 can be used straightforwardly. For example, in [6.6] $r(x,y)$ is clipped so that its absolute value does not exceed 40 out of a possible 255. This clipping causes errors only near sharp edges where it is much less visible. Next, the entire $r(x,y)$ is transform coded using the RDFT (see Sec. 5.3.1) with uniform quantization and a frequency weighted MSE distortion criterion. A single variable word-length code is then used for coding each of the

---

* Gamma corrected and digitized by non-interlaced raster scanning and 8-bit PCM quantization.

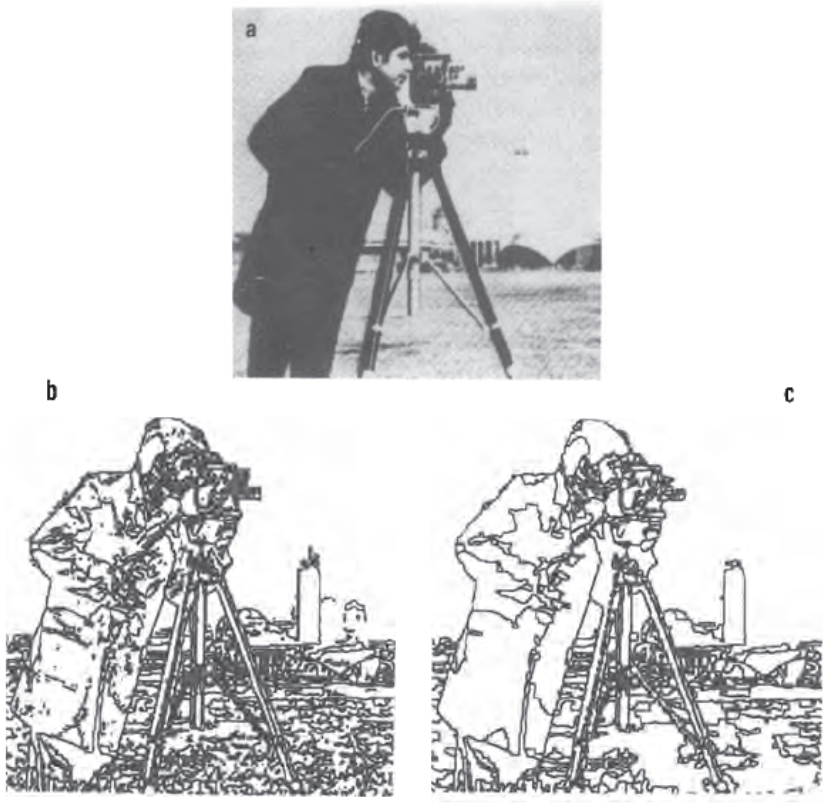


Fig. 6.5.1 (a) Original Image. (b) Example of contours of thresholded spatial derivative. (c) Remaining objects after elimination of unclosed contours and small objects (from Kocher and Kunt [6.7]).

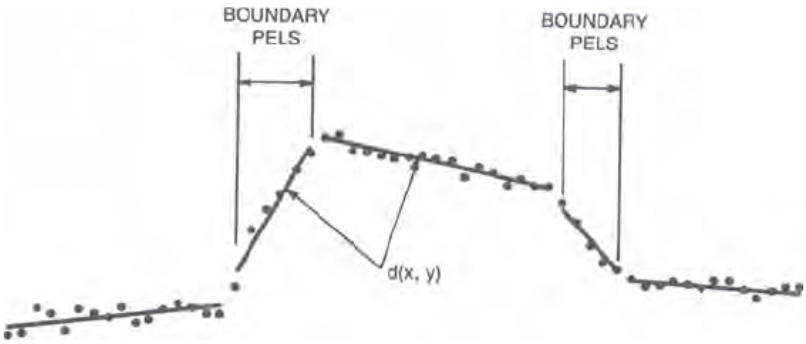


Fig. 6.5.2 Piecewise linear approximation to pel values along a scan line. The resulting discontinuous function is used to form $d(x,y)$.

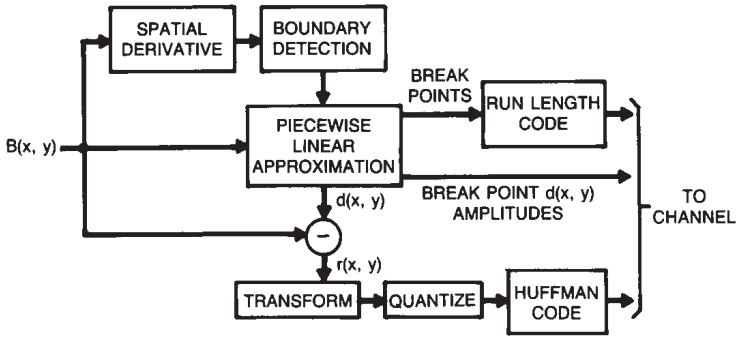


Fig. 6.5.3 System for coding images by decomposing them into edges and texture. For simplicity, only the luminance coding is shown.

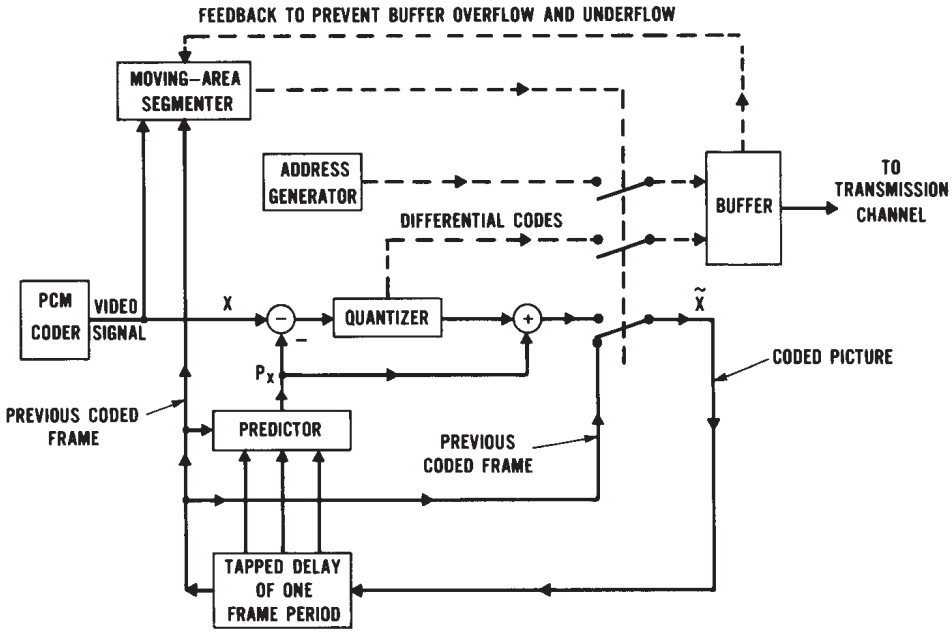


Fig. 6.6.1 Conditional replenishment coder that sends moving-area picture elements (pels) via predictive coding. For each pel X, the predictor uses previously coded pels to compute a prediction P_x. The differential signal $X-P_x$ is quantized and coded. If X is a moving-area pel as determined by the segmenter, the differential code plus addressing information is fed to the buffer to await transmission, and a new pel value $\tilde{X}$ is passed to the frame memory to be used later by the coder.

transmitted coefficients. Using this algorithm, which is shown diagrammatically in Fig. 6.5.3, images have been coded with moderate quality at rates around 0.8 bits/pel.

For a fixed picture quality, image decomposition achieves a lower bit-rate than using transform coding by itself. However, the resulting bit-rates will vary markedly from picture to picture. If a fixed bit-rate is desired for both $r(x,y)$ and $d(x,y)$, then the complexity of image decomposition rises significantly since the edge detection algorithm used to define $d(x,y)$ must then adapt itself to produce an approximately constant number of breakpoints. An alternative would be to use a nonadaptive edge detection algorithm, accepting whatever bit-rate results for $d(x,y)$ and then simply assigning whatever bits are left to the coding of $r(x,y)$. However, achievable picture quality would not be as high.

6.6 Conditional Replenishment DPCM Codecs

These systems typically have the generic form shown in Fig. 6.6.1. Moving-area pels (as defined by the segmenter) are transmitted via predictive coding, whereas stationary-area pels are not transmitted at all. Instead their previous frame values are repeated both at the transmitter and receiver. The quantizer might be a scalar quantizer operating on individual pels, or it might be a block quantizer operating on two-dimensional blocks of pels either in the pel domain or in some transform domain. The predictor might be a simple previous frame predictor, or it might be scene adaptive incorporating such algorithms as motion compensation. With highly efficient predictors, a further segmentation may be utilized that classifies moving-area pels as either *predictable*, in which case the prediction error is nearly zero and need not be sent, or *unpredictable*, in which case prediction errors must be transmitted.

The data rate produced by the coder is directly related to the amount of movement in the scene. As movement increases, the buffer would normally tend to fill unless steps are taken to reduce the data rate. These include spatial subsampling, i.e., only sending a subset of the moving-area pels, coarser quantization in order to reduce the number of bits per transmitted pel, and field/frame repeating, i.e., sending nothing for an entire field/frame period.

In many countries of the world, digital transmission facilities abound at certain *primary* bit-rates. In North America and Japan, one of the most plentiful is the so called DS1 rate of 1.544 Mbits/sec. In Europe, the corresponding rate is 2.044 Mbits/sec. Thus, there is great incentive to use these fairly ubiquitous bit-rates for videoconferencing or videotelephone services, and indeed standards have already been set by the CCITT[6.8] for these rates.

In videoconferencing applications the television camera is fixed. Picture data is then produced only if there are moving objects in the scene. In this case the amount of moving-area in the picture is generally much smaller than the amount of stationary area, and significant bit-rate reduction is possible using conditional frame replenishment. In those rare instances where there is a lot of

movement, the data rate generated by the coder can be lowered and buffer overflow prevented by reducing the moving-area resolution (either spatial, temporal, or amplitude) and keeping the stationary background area at full resolution.

Some videoconferencing systems have several cameras, and at any given time the one pointing toward the current or most recent speaker is switched into the transmission. In this case, data is produced also by scene changes when the camera is switched. However, we have seen in Chapter 4 that a new full-resolution picture need not be sent in one frame period. Viewers cannot perceive full resolution for several frame periods after a scene change. Thus, a frame replenishment coder need only transmit a very low-resolution picture immediately following a scene change. If full resolution can be built up within a reasonable time (8-10 frame periods), picture degradation will not be objectionable.

For purposes of illustration in this section, we will excerpt design details from one of the CCITT algorithms[6.8], in particular, the "Inter-Regional Part 2" codec, which can input either 525-line or 625-line video and operate at either 1.544 Mbits/sec or 2.044 Mbits/sec. This algorithm is a compromise design that accomplishes compatibility between the industrial countries of the world and can be implemented with relative simplicity. For this reason, its picture resolution is somewhat less than might be accomplished if compatibility and simplicity were sacrificed

The input interlaced video consists of luminance and chrominance *Y Cb Cr* components as defined in Section 2.2.5. These are sampled with 8-bits per pel at a sampling rate (blanking included) of 320 pels per line for luminance and 64 pels per line for chrominance. Pel amplitudes are restricted to the range 16–239. Next, a spatial averaging process is carried out to reduce the number of active lines per field to 143. The chrominance is then line interleaved as in Section 2.2.5, so that only one chrominance component per line need be transmitted. This results in 256 active luminance pels and 52 active chrominance pels per line, with an uncompressed bit-rate of 17.6 Mbits/second. Interpolation at the receiver restores the correct number of lines for display.

The codec is capable of operating at frame rates of 25 Hz or 30 Hz. If the input and output frame rates are different, it transmits 25 frames per second. With a 30 Hz input frame rate, this is accomplished by not coding one out of six frames. With a 30 Hz output, a separate frame memory is used to convert from 25 Hz by means of temporal interpolation.

We saw from Fig. 3.2.12b that among the simple linear interframe predictors, the element difference of field difference had the lowest entropy, with element difference of frame difference and line difference of frame difference coming second. At higher speeds of movement intrafield predictors such as the simple element difference or the intrafield $(A+D)/2$ predictor of Fig. 6.3.1 also perform well.

However, with most interframe codecs horizontal and vertical spatial subsampling of the moving area is an important technique for data rate reduction when buffer overflow threatens (with interlace vertical subsampling is equivalent to sending moving-area pels only in alternate fields). Moreover, with subsampling, the moving-area pels that are not transmitted are interpolated from their neighbors, and in this case, the predictor should only use non-subsampled pels that are available with fairly good accuracy. Otherwise, the prediction values may be corrupted by interpolation error.

During intervals of low movement when little data is being generated, predictor efficiency is of less concern than during periods of high movement. Since intrafield and interfield predictors perform about the same during high movement, the inter-regional CCITT codec uses a relatively simple intrafield prediction to code moving-area pels. For luminance pels the $(A+D)/2$ predictor is used, whereas for chrominance, pel A is used as a prediction. During periods of horizontal subsampling, if either pel A or pel D were obtained by interpolation, then the non-interpolated pel to the left is used instead in the prediction. In normal operation, prediction errors are quantized to 16 levels and coded with variable word-lengths as shown in Fig. 6.6.2.

The inter-regional codec uses adaptive horizontal subsampling to reduce the data rate when necessary. Normally, even numbered pels are sent during even lines, and odd pels during odd lines. Transmitted prediction errors are quantized and coded with 8 levels, as shown in Fig. 6.6.3. However, some of the intervening pels may also be sent in order to reduce the interpolation error. Differential values for these pels are coded using the "extra" variable word-length codes given in Fig. 6.6.3. Pels that are horizontally subsampled are obtained by interpolating the two horizontally adjacent pels.

As mentioned above, vertical subsampling results in not sending anything for an entire field period. In principle, all pels of the field could be replaced by an average of pels in the adjacent fields. For example, in Fig. 6.3.1 each pel X could be replaced by a four-way average of pels F and I in the two adjacent fields. However, this would cause a noticeable resolution reduction in non-moving areas of the picture. Thus, the inter-regional codec only performs the interpolation of pel X if all four temporally nearby pels (F and I in the adjacent fields) were nonmoving. Otherwise, pel X is repeated from two fields back, thus maintaining full spatial resolution in the stationary background.

If spatio-temporal subsampling is to be used as a data reduction mechanism then, in principle, we should low-pass filter the signal in order to avoid aliasing due to too low a sampling rate. Spatial filtering is very difficult to implement only in the moving area. However, temporal filtering is not, and moreover it has almost the same effect. Fig. 6.6.4(a) shows a one-stage recursive temporal filter where the value of α determines the amount of filtering, and the frame memory is the same one used by the coder (see Fig. 6.6.1). Fig. 6.6.4(b) shows a temporal filter containing its own frame memory. In this case, coding error does

Input Levels	Output Levels	Variable-length Code	Code No.
−255 to −125	−141	1 0 0 0 0 0 0 0 0 1	17
−124 to −95	−108	1 0 0 0 0 0 0 0 1	16
−94 to −70	−81	1 0 0 0 0 0 0 1	15
−69 to −49	−58	1 0 0 0 0 0 1	14
−48 to −32	−39	1 0 0 0 0 1	13
−31 to −19	−24	1 0 0 0 1	12
−18 to −9	−13	1 0 1	10
−8 to −1	−4	1 1	9
0 to 7	+3	0 1	1
8 to 17	+12	0 0 1	2
18 to 30	+23	0 0 0 1	3
31 to 47	+38	0 0 0 0 1	4
48 to 68	+57	0 0 0 0 0 1	5
69 to 93	+80	0 0 0 0 0 0 1	6
94 to 123	+107	0 0 0 0 0 0 0 1	7
124 to 255	+140	0 0 0 0 0 0 0 0 1	8

Fig. 6.6.2 Quantization parameters and code words for prediction errors during full horizontal sampling. The end-of-cluster (EOC) code number 11 is 1001.

Quantization		Variable-Length Codes			
Input Range	Output Levels	Normal Elements	Code No.	Extra Elements	Code No.
−255 to −41	−50	10000001	15	1000000001	17
−40 to −24	−31	100001	13	100000001	16
−23 to −11	−16	101	10	1000001	14
−10 to −1	−5	11	9	10001	12
0 to +9	+4	01	1	0001	3
10 to 22	+15	001	2	000001	5
23 to 39	+30	00001	4	00000001	7
40 to 255	+49	0000001	6	000000001	8

Fig. 6.6.3 Quantization parameters and code words during horizontal subsampling. Intervening pels can be sent, if desired, using the *extra* code words.

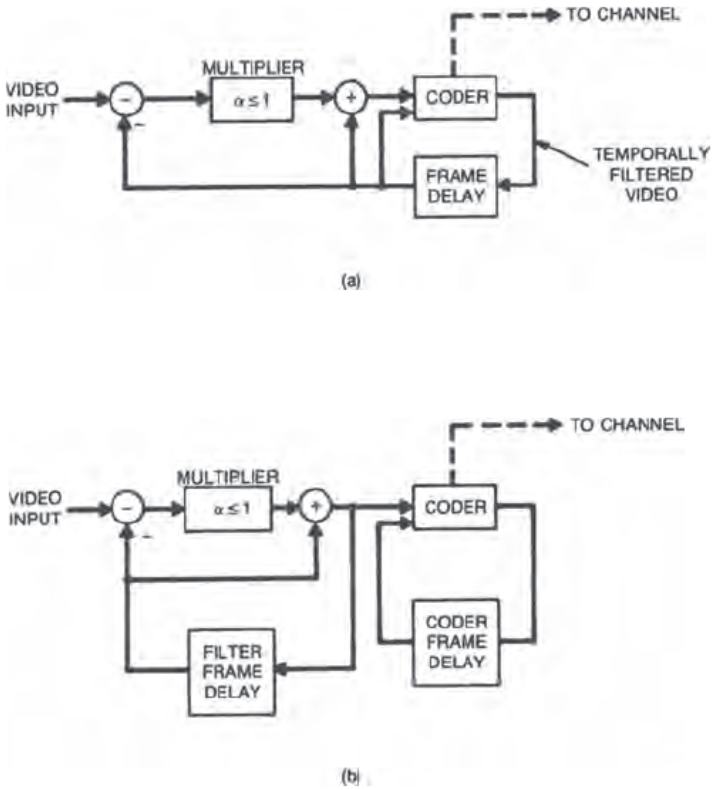


Fig. 6.6.4 Interframe coder preceded by a single-pole temporal filter. The lower the value of α, the more the temporal filtering. In (a) the filter and coder share the same frame memory. In (b) the temporal filter has its own memory.

not propagate through the temporal filter. Additional memories, i.e., a higher order filter, give improved performance, but at a higher cost. Also, replacing the constant gain multiplier by a nonlinear characteristic having lower gain for small input and larger gain for large input can still reduce the input noise, yet not cause as much blurring of the moving areas as occurs with a constant gain. Temporal filtering not only make subsampling effects less visible, it also reduces the level of frame-to-frame noise in stationary-areas thus making moving-area segmentation much easier. In addition, because temporal filtering causes blurring in the moving-area, it reduces the entropy of the differential signal — especially for intrafield predictors such as the ones used by the inter-regional codec.

An additional problem for vertical and horizontal subsampling is that the high data rates requiring subsampling may be due to either relatively small areas moving rapidly or large areas moving slowly. In the latter case, subsampling degradation is easily visible since temporal filtering (either camera integration or external) does not reduce the spatial resolution sufficiently to mask the subsampling effects.

The codec has the capability of sending entire lines as 8-bit PCM instead of as DPCM. This is called "forced updating" and is necessary to accommodate transmission bit errors. It is also convenient in preventing buffer underflow.

Transmission of each field starts with a 28-bit field start code indicating whether it is an even field or an odd field. Vertical subsampling is signaled simply by a missing field start code. Transmission of each line starts with a 20-bit line-start code that also indicates whether or not horizontal subsampling is to be used. Clusters of moving-area luminance pels, as defined by the segmenter, are then sent, followed by clusters of moving-area chrominance pels.

Each cluster begins with the 8-bit PCM value of the first pel of the cluster followed by its 8-bit address along the line. Then comes the coded DPCM data of the remaining pels of the cluster, followed finally by the end of cluster (EOC) code. Just before the first chrominance cluster of the line, an 8-bit color escape word 00001001 is sent, which is outside the range of allowable PCM values and therefore distinguishable from them. Signaling of an entire line of PCM values is accomplished with 11111111, another escape code.

The coder buffer size is chosen as 96 kbits, which is small enough to assure a one-way delay of less than 200 msec.

Adaptive moving-area resolution control is used to keep the generated data rate more constant than it would be otherwise, and therefore avoid requiring extremely large buffer memories (and therefore large delay) or overflowing buffers of more practical size. The most convenient measure of whether or not moving-area resolution should be altered is the likelihood of near term buffer overflow or underflow. Typically, if the buffer is filling rapidly, resolution is decreased, and vice versa if the buffer is nearly empty. This measure is not an instantaneous indication of movement. However, if the buffer is of moderate

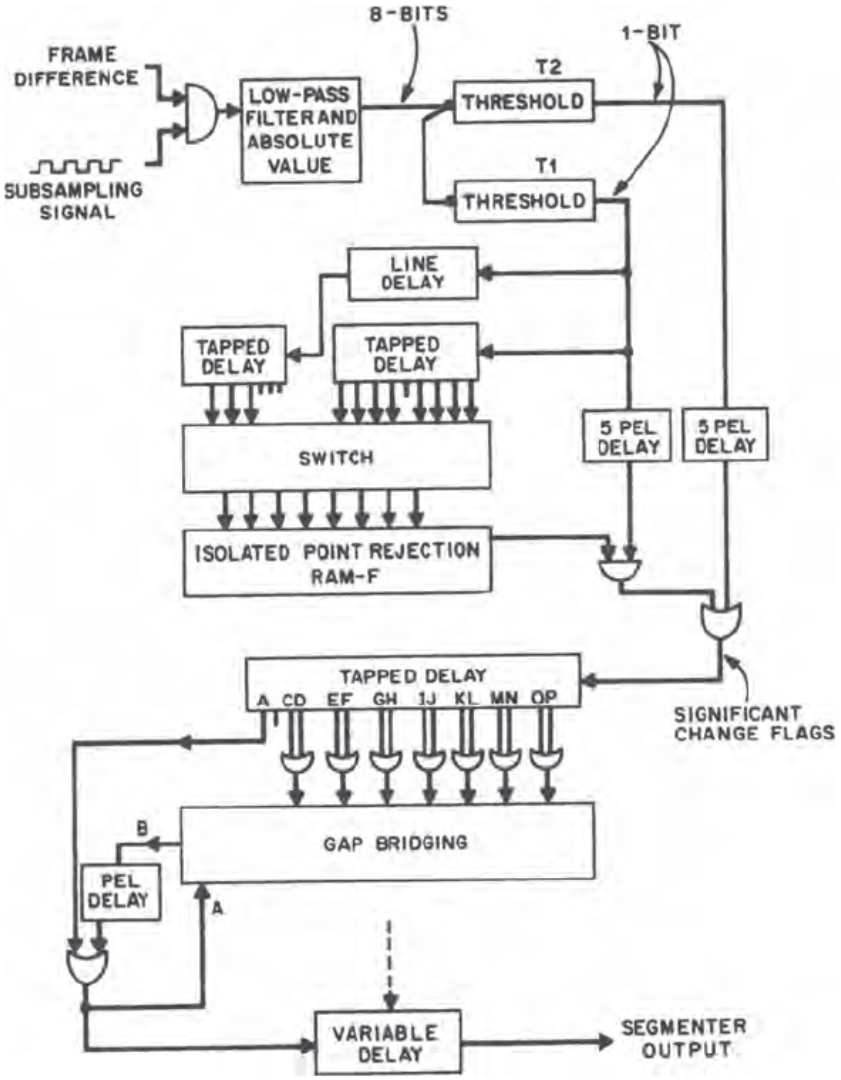


Fig. 6.6.5 A segmenter determines which pels of the frame are *moving* and which are *stationary background*. Frame difference magnitudes larger than $T2$ indicate *moving* pels. Frame difference magnitudes between $T1$ and $T2$ that are isolated are assumed to be due to noise, and do not indicate moving pels.

size, e.g., less than a few frames of transmitted data, the time lag is not a serious handicap. A possible serious drawback of using this measure is that it is difficult to distinguish between small areas moving fast and large areas moving slowly.

It is not known how to optimally control moving-area resolution for a given channel bit-rate and buffer size. Most resolution control algorithms have been designed on an ad hoc basis by trial and error. Reducing resolution in the moving area as buffer fullness exceeds certain thresholds gives reasonably good results as long as some hysteresis is built in to avoid oscillations between different resolutions. Without hysteresis it is possible for the top and bottom of a picture to be displayed with different resolutions, which may be objectionable if the difference is visible. A simple technique for avoiding oscillations is to reduce the moving area resolution when the buffer queue length exceeds a certain threshold. However, only when the queue length falls below a much lower threshold is the moving-area resolution increased again. Alternatively, mode-switching could be done only in the vertical blanking interval.

What follows is an example of how to combine these techniques in a multimode conditional replenishment codec[6.9,6.10] for coding a 525 line, component color picture at 1.5 Mb/s (0.19 bits/pel at 8 MHz sampling rate).

Segmenting in such a system is fairly straightforward as shown in Fig. 6.6.5. The luminance frame difference is low-pass filtered and threshold detected with two alterable thresholds, $T2 \geq T1$. Differences larger than $T2$ are immediately classified as significant, whereas differences exceeding $T1$ must undergo isolated point rejection before being classified. Following this, small gaps between clusters of changes are redefined as *changed* to increase addressing efficiency. A variable amount of temporal filtering is also included to reduce entropies of differential signals, noise on the input signal and quantization noise.

Nine modes (0-8) are used in coding both the luminance and chrominance as shown in Fig. 6.6.6. In general, switching to the next higher (movement) mode takes place as soon as the buffer queue length exceeds an upper limit specified for that mode. However, switching to a lower (movement) mode only occurs at the end of a coded field if at that time the buffer queue length is below a lower limit specified for that mode. This avoids oscillations between modes, which otherwise could cause unnecessary impairment and needless filling of the buffer.

In the lower modes, where relatively high resolution is maintained, 16-level quantization of the differential signal is used. However, for the higher modes 8-level quantization is used. Variable word-length coding is used for both quantizers, with one 4-bit word reserved for end-of-cluster signaling. Varying the quantization and temporal filtering requires that changes be made in the segmenting, as well. Thus, different modes have different frame difference low-pass filters and different thresholds. Subsampling is utilized straightforwardly under buffer control in the following order: vertical 2:1 (field

Mode	Buffer Threshold (bits) Lower	Upper	Number of Quantizer Levels	Sub Sampling 2:1 Vert	2:1 Horiz	Comments
0	0	6400	256	No	No	8-bit PCM
1	1600	38400	16	No	No	Low Threshold T1, T2
2	1600	38400	16	No	No	Higher T1, T2
3	3200	38400	16	Yes	No	Alternate Fields Dropped
4	3200	51200	16	Yes	No	More Temporal Filtering
5	3200	54400	8	Yes	Yes	Adaptive Horiz. Subsamp.
6	10000	70400	8	Yes	Yes	Horiz. Subsamp. All Pels
7	32000	83200	8	Yes	Yes	More Temporal Filtering
8	32000	96000	0			Drop Frames

Fig. 6.6.6 Modes used by the coder are controlled by the fullness of the buffer.

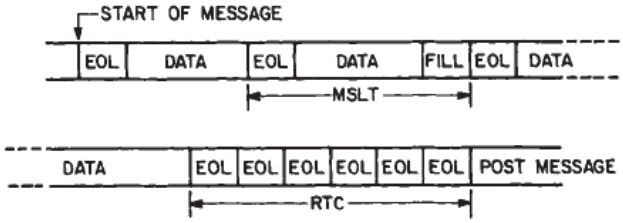


Fig. 6.7.1 Format of transmitted messages for several lines. Minimum scan line time (MSLT) is obtained by using fill bits as shown on the top line. End of the document transmission is signalled by 6 consecutive end of line (EOL) codewords, which indicates return to control (RTC).

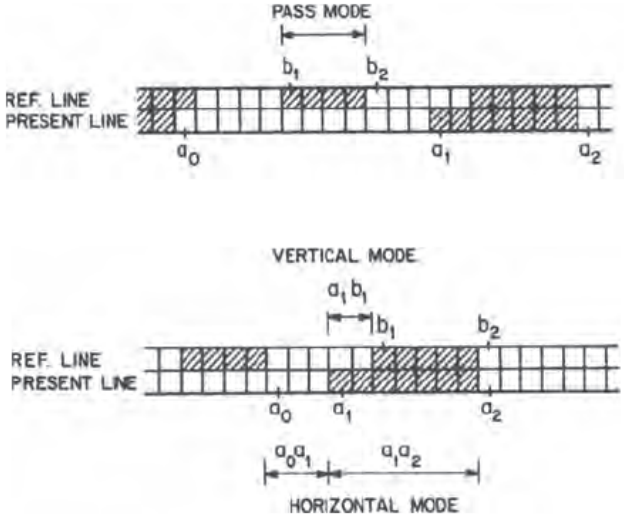


Fig. 6.7.2 Examples of coding modes for the two dimensional coding scheme. Top configuration shows the pass mode and the bottom one shows the horizontal and vertical modes. (from Hunter and Robinson [2.6.2])

dropping), horizontal 2:1, frame repeating. The last is used only during camera movement or scene changes.

6.6.1 Error Control

With intraframe coding where each frame is transmitted separately, the effect of a digital transmission error lasts for at most a frame period. With interframe coding such effects can last much longer. In general, the more one removes redundancy from a television signal through the use of sophisticated coding techniques, the more important each bit of transmitted information becomes, and the more noticeable are the effects of digital transmission errors. For example, with simple frame differential transmission of every pel, an error appears in only one pel and lasts indefinitely unless some error correction strategy is employed, e.g., incorporating leak into the feedback loop as described in Sec. 5.2.3.[6.11,6.12] With frame replenishment, leak is not very effective since not all pels are sent in every frame. Furthermore, if higher order DPCM and/or variable word-length coding are used, defects due to transmission errors can spread beyond one pel, possibly into adjacent lines.

Forward acting error control has been incorporated into frame replenishment coders in which effects due to an error are confined for the most part to one line. Channel error correcting codes might be used such as BCH or convolutional codes.[6.13] Special protection of sync and cluster demarcation words is also worthwhile in order to minimize the area affected by a transmission error.[6.14] In spite of these precautions, however, errors are bound to appear in the pictures sooner or later, and further steps must be taken to deal with them.

Typically, one or more lines or blocks per frame are sent via PCM or some other intraframe coding method that does not rely on pels from surrounding lines or blocks. This is called *forced updating*.[6.13] In this way, in a few seconds, every pel in the frame can be updated with correct values, thus wiping out any residual effects due to errors that survive the channel error correcting codes.

Several seconds is a long time for a highly visible error defect to remain at the display. Through the use of highly reliable error detection codes, it is possible for the receiver to tell which lines contain errors. If these lines are then replaced, for example, with an average of adjacent lines, then the effects of errors (if their rate is reasonably low) can be made much less visible.[6.12] This technique usually requires that once an erroneous line is replaced by an average, replacement of that line must continue during succeeding frames until PCM values arrive for that line.

Error detection followed by a request for retransmission is often not feasible for several reasons. First, for the high data rates usually entailed in video transmission, very large buffers would be required. Second, for the long distances often involved, too much delay would be introduced for two-way video communication. However, if substitution of an erroneous line, as

described above, were followed by a request for PCM transmission of that line during some later frame, then the duration of error effects could be markedly reduced compared with strictly forward acting techniques.

If the effects of a digital transmission error propagate to adjacent lines, as they would for example with DPCM that uses pels from previous lines, then line substitution does not work. Substitution from the previous frame would not be very disturbing if movement were slow. However, in rapidly moving areas, picture breakup would occur, and substitution of the entire field would probably be preferable. Subdivision of the picture into smaller blocks that are coded independently of one another might alleviate the situation somewhat. In any event, substitution could not be carried out for very long — one or two frames at most. A request for retransmission would have to be issued to minimize the visibility of effects due to transmission errors.

6.7 Graphics Coders

In this section we give details of two coding schemes standardized by the CCITT* along with their performance on the CCITT pictures. These pictures have already been described in Chapter 2 and are shown in Fig. 2.6.6. They have played a key role in optimizing the parameters of the adopted standard coding schemes. The first coding scheme uses one-dimensional run-length coding, whereas the second exploits two-dimensional correlation in the data. These schemes, called Group 3[6.15] (or G3) schemes, are used primarily for transmitting a A4 size (210 mm × 298 mm) document over a public switched telephone network. Since the telephone network is prone to transmission errors, two forms of redundancy are provided to avoid degradation of the image when errors occur. These include a special code at the end of each line and transmission of one-dimensional compressed code every two or four lines.

The CCITT has also developed recommendations called Group 4[6.16] (or G4), applicable for more general pictures and communications media. As examples of this, for digital communication networks containing error control, G4 schemes are obtained from G3 schemes by removing the above two forms of redundancy (i.e., end of line code and periodic transmission of one-dimensional compressed data) since the network provides error free transmission. Also, G4 schemes allow more general pictures by incorporating four possible resolutions: 200, 240, 300 and 400 pels per inch. A mixed mode operation containing symbols and graphics is also allowed. Those parts of the image containing alphanumerics are transmitted as characters using formats such as ASCII, whereas other parts of the image containing non-character information such as

* Now known as ITU-T.

Table 6.7.1 Modified Huffman code for the run-lengths in the one-dimensional coding scheme. Both terminating codes (TC) and make-up codewords (MUC) are shown.

Terminating Codewords (TC)

Run-length	White Runs	Black Runs
0	00110101	0000110111
1	000111	010
2	0111	11
3	1000	10
4	1011	011
5	1100	0011
6	1110	0010
7	1111	00011
8	10011	000101
9	10100	000100
10	00111	0000100
11	01000	0000101
12	001000	0000111
13	000011	00000100
14	110100	00000111
15	110101	000011000
16	101010	0000010111
17	101011	0000011000
18	0100111	0000001000
19	0001100	00001100111
20	0001000	00001101000
21	0010111	00001101100
22	0000011	00000110111
23	0000100	00000101000
24	0101000	00000010111
25	0101011	00000011000
26	0010011	000011001010
27	0100100	000011001011
28	0011000	000011001100
29	00000010	000011001101
30	00000011	000001101000
31	00011010	000001101001
32	00011011	000001101010
33	00010010	000001101011
34	00010011	000011010010
35	00010100	000011010011
36	00010101	000011010100
37	00010110	000011010101
38	00010111	000011010110
39	00101000	000011010111
40	00101001	000001101100
41	00101010	000001101101
42	00101011	000011011010
43	00101100	000011011011
44	00101101	000001010100
45	00000100	000001010101

Terminating Codewords (TC) Con't.

Run-length	White Runs	Black Runs
46	00000101	000001010110
47	00001010	000001010111
48	00001011	000001100100
49	01010010	000001100101
50	01010011	000001010010
51	01010100	000001010011
52	01010101	000000100100
53	00100100	000000110111
54	00100101	000000111000
55	01011000	000000100111
56	01011001	000000101000
57	01011010	000001011000
58	01011011	000001011001
59	01001010	000000101011
60	01001011	000000101100
61	00110010	000001011010
62	00110011	000001100110
63	00110100	000001100111
	Make-up Codewords (MUC)	
64	11011	0000001111
128	10010	000011001000
192	010111	000011001001
256	0110111	000001011011
320	00110110	000000110011
384	00110111	000000110100
448	01100100	000000110101
512	01100101	0000001101100
576	01101000	0000001101101
640	01100111	0000001001010
704	011001100	0000001001011
768	011001101	0000001001100
832	011010010	0000001001101
896	011010011	0000001110010
960	011010100	0000001110011
1024	011010101	0000001110100
1088	011010110	0000001110101
1152	011010111	0000001110110
1216	011011000	0000001110111
1280	011011001	0000001010010
1344	011011010	0000001010011
1408	011011011	0000001010100
1472	010011000	0000001010101
1536	010011001	0000001011010
1600	010011010	0000001011011
1664	011000	0000001100100
1728	010011011	0000001100101
EOL	000000000001	000000000001

drawings and handwriting are coded by graphics coding schemes. In any case, the graphics coding schemes used for G4 are minor variations of the basic G3 schemes that are described in detail below.

Another scheme, described in Chapter 7, was developed by the ISO Joint Bilevel Image Group (JBIG). It is more efficient than G3/G4 and also has the capability of progressive coding.

6.7.1 CCITT G3/G4 — One-Dimensional Coding Scheme

Each scan line is coded by assigning words to the runs of "white" and "black" elements that alternate along a scan line. The lines are assumed to begin with a white run; if the first actual run on a line is black, then a white run of zero length is assumed and the corresponding code word is transmitted at the beginning of the line. Separate code tables based on a modified Huffman procedure using the statistics of the eight CCITT pictures are used to represent the black and white runs. The code table is given in Table 6.7.1. The code table contains codes for run-lengths up to 1728 pels (length of a scan line). The code words are of two types: terminating codes (TC) and make-up codes (MUC). Run-lengths between 0 and 63 are transmitted using a single terminating codeword. Run-lengths between 64 and 1728 are transmitted by a MUC followed by a TC. The MUC represents a run-length value of $64 \times N$ (where N is an integer between 1 and 27), which is equal to, or shorter than, the value of the run to be transmitted. The difference between the MUC and the actual value of the run-length is specified by the trailing TC. Each coded line is followed by the end-of-line (EOL) codeword, which is a unique sequence that cannot occur within a valid line of coded data. The codeword chosen for EOL is 000000000001, eleven 0's followed by a 1. If the number of coded bits in a line is fewer than a certain minimum value, then "fill" bits consisting of strings of 0's of varying length are inserted between the line of coded data and the EOL codeword. This ensures that each coded scan line will require more than a specified amount of transmission time so that transmitters and receivers can keep in step, and mechanical limitations with respect to scanning or printing can be overcome. The recommended standard minimum is 96 bits per line (at 4800 bits/sec transmission rate) with options for 48, 24 and 0 bits. The fill bits are easily recognized and discarded by the receiver. Fig. 6.7.1 shows the format of the data for several coded lines. The end of document transmission is indicated by six consecutive EOL codewords that form the return control signal. The compression performance for this run-length coding scheme is given in Table 6.7.2 for the eight CCITT documents. The average compression factor is 9.7.

6.7.2 CCITT — Two-Dimensional Coding Scheme

This scheme is known as the "modified READ code" (Relative Element Address Designate), and has evolved through the various proposals submitted to

Table 6.7.2 Performance and the relevant statistics of the one dimensional coding scheme for the eight CCITT documents. Maximum compression ratio is computed by Eq. (3.4.9) using the measured entropies of black and white runs for each document. Compression ratio is computed using the actual modified Huffman code, but without end of line and fill bits. (from Hunter and Robinson [2.6.2])

CCITT DOCUMENT NO.	AVERAGE WHITE RUN-LENGTH	AVERAGE BLACK RUN-LENGTH	ENTROPY OF WHITE RUNS	ENTROPY OF BLACK RUNS	MAXIMUM COMPRESSION RATIO	COMPRESSION RATIO
1	134.6	6.790	5.230	3.592	16.02	15.16
2	167.9	14.02	5.989	4.457	17.41	16.67
3	71.50	8.468	5.189	3.587	9.112	8.350
4	36.38	5.673	4.574	3.126	5.461	4.911
5	66.41	6.966	5.280	3.339	8.513	7.927
6	90.65	8.001	5.063	3.651	11.32	10.78
7	39.07	4.442	5.320	3.068	5.188	4.990
8	64.30	60.56	4.427	5.310	11.52	8.665

Table 6.7.3 Code table for the two-dimensional coding scheme. $M(a_0a_1)$ and $M(a_1a_2)$ are codewords for the appropriate run-lengths taken from Table 6.7.1. (from Hunter and Robinson [2.6.2])

MODE	CHANGING ELEMENTS TO BE CODED		NOTATION	CODEWORD
PASS	b_1, b_2		P	0001
HORIZONTAL	a_0a_1, a_1a_2		H	$001+M(a_0a_1)+M(a_1a_2)$
VERTICAL	a_1 JUST UNDER b_1	$a_1b_1=0$	V(0)	1
	a_1 TO THE RIGHT OF b_1	$a_1b_1=2$	$V_R(2)$	000011
		$a_1b_1=1$	$V_R(3)$	0000011
	a_1 TO THE LEFT OF b_1	$a_1b_1=1$	$V_L(1)$	010
		$a_1b_1=2$	$V_L(2)$	000010
		$a_1b_1=3$	$V_L(3)$	000010
END-OF-LINE CODEWORD (EOL)				000000000001
1-D CODING OF NEXT LINE 2-D CODING OF NEXT LINE				EOL + '1' EOL + '0'

the CCITT until 1979. It is a line-by-line scheme in which the position of each *changing* element on the *present line* is coded with respect to either the position of a corresponding changing element on the *reference line*, which lies immediately above the present line, or with respect to the preceding changing element on the present line. After the present line has been coded, it becomes the reference line for the next line.

The changing element is an element of different color from that of the previous element along the same line. The coding procedure uses five changing elements defined as in Fig. 6.7.2.

1. a_0 : the first changing element on the present line.
2. a_1 : the next changing element on the present line; by our definition, it has opposite color to a_0 and gets coded next.
3. a_2 : the changing element following a_1 on the present line.
4. b_1 : the changing element on the reference line to the right of a_0 with the same color as a_1.
5. b_2 : the changing element following b_1 on the reference line.

Depending on the relative position of the changing element that is being coded, the coder operates in three modes.

(i) Pass Mode
In this mode, the element of b_2 lies horizontally to the left of a_1. Using our definitions above, it occurs whenever the white or black runs on the reference line are not adjacent to corresponding white or black runs on the present line. The pass mode is represented by a single codeword.

(ii) Vertical Mode
In this mode, the element a_1 is sufficiently close to b_1 and is therefore coded relative to the position of b_1. It is used only if a_1 is to the left or right of b_1 by at most 3 pels and hence, the relative distance a_1b_1 can take on any of seven values $V(0)$, $V_R(1)$, $V_R(2)$, $V_R(3)$, $V_L(1)$, $V_L(2)$, $V_L(3)$. The subscript R is used if a_1 is to the right of b_1. Subscript L is used otherwise. The number in parentheses indicates the distance a_1b_1 in pels.

(iii) Horizontal Mode
If a_1 is not sufficiently close to b_1, then its position must be coded by horizontal mode. Thus, the run-lengths a_0a_1 and a_1a_2 are coded using the concatenation of three codewords H, $M(a_0a_1)$ and $M(a_1a_2)$. Codeword H, taken to be 001, serves as a prefix or flag, and $M(a_0a_1)$ and $M(a_1a_2)$ are taken from the code tables to represent the colors and values of the run-lengths a_0a_1 and a_1a_2. The Table is based on the modified Huffman procedure as in CCITT scheme 1.

Table 6.7.4 Performance of the two-dimensional coding schemes for low and high resolutions. Actual coded bits are shown for each of the CCITT documents for $K = 2$, 4 and ∞.

CCITT DOCUMENT NUMBER	LOW RESOLUTION (100 PELS/INCH)	HIGH RESOLUTION (200 PELS/INCH)		
	K = 2	K = 4	K = ∞	COMPRESSION RATIO
1	130684	207660	175704	21.6
2	106851	175163	117304	32.4
3	207584	326297	260527	14.6
4	408261	654436	585074	6.5
5	226285	353172	288655	13.2
6	150572	225879	164085	23.2
7	402333	651643	585135	6.5
8	184369	264029	183674	20.7
AVERAGE	227117	355034	295019	12.9

In practice, the coder selects first the coding mode and then the appropriate codeword from Table 6.7.3. The two steps are:

Step 1

If b_2 is detected before a_1 then the pass mode is selected and the codeword 0001 is transmitted. The reference pel a_0 is set on the pel below b_2 in preparation for the next coding. If a pass mode is not detected, step 2 is followed.

Step 2

The horizontal distance between a_1 and b_1 is determined. If the distance is at most three, then vertical mode is used, and a_0 is set on the position of a_1 in preparation for the coding of the next changing element. Otherwise, the horizontal mode is used.

The performance of the two-dimensional code is given in terms of the number of coded bits for each CCITT document in Table 6.7.4. Use of vertical correlation makes it possible for the effects of any transmission error to propagate vertically down. To combat this, every Kth line is coded using the one-dimensional code. At the resolution of 100 pels/inch, with $K=2$, at a transmission rate of 4800 bits/sec, the average transmission time is 47.3 sec, whereas at high resolution with $K=4$ the average transmission time is 74 sec. At $K=\infty$ and high resolution, the average transmission time is 61.5 sec. Thus, there is a significant penalty for using the one-dimensional code every fourth line. However, in general, for high resolution there is a considerable decrease in transmission time by using the two-dimensional code as compared to the one-dimensional code.

References

6.1 R. C. Brainard and J. H. Othmer, "VLSI Implementation of a DPCM Compression Algorithm for Digital TV," *IEEE Trans. Communications,* August 1987, pp. 854-856.

6.2 N. S. Jayant and P. Noll, *Digital Coding of Waveforms*, Chapter 6, Prentice Hall, New York, 1984.

6.3 R. C. Brainard and A. N. Netravali, "Digital Broadcast TV at 45 Mb/s," GLOBECOM '82 Proceedings, Miami 1982, pp. B6.1.1-4. Also, J. SMPTE.

6.4 H. Murakami, et al., "A 15 Mbit/s Universal Codec for TV signals Using a Median Adaptive Prediction Coding Method," Sixth Int. Conf. Digital Satellite Communications, September 19-23, 1983, Phoenix, Arizona, pp. VII-A-17 to VII-A-24.

6.5 F. A. Kamangar and K. R. Rao, "Interfield Hybrid Coding of Component Television Signals," *IEEE Trans. Communications,* COM-29, December 1981, pp. 1740-1753.

6.6 J. K. Yan and D. J. Sakrison, "Encoding of Images Based on a Two-Component Source Model," *IEEE Trans. Commun.,* COM-25, November, 1977, pp. 1315-1322.

6.7 M. Kocher and M. Kunt, "Image Data Compression by Contour Texture Modelling", *Proceedings of SPIE,* v. 397, April 19-22, 1983, Geneva, pp. 132-139.

6.8 International Telegraph and Telephone Consultative Committee (CCITT) Recommendations H.110, H.120, and H.130; Parts 1 to 3.

6.9 B. G. Haskell, et al., "Interframe Coding of 525-Line Monochrome Television at 1.5 Mbits/s," *IEEE Trans. Commun.,* COM-25, November, 1977, pp. 1339-1348.

6.10 T. S. Duffy and R. C. Nicol, "A Codec for International Visual Teleconferencing," Globecom '82 Conference Record, v. 3, December, 1982, Miami, pp. E2.5.1-.5.

6.11 D. J. Connor, R. C. Brainard, and J. O. Limb, "Intraframe Coding for Picture Transmission," *Proc. IEEE,* 60, No. 7, July, 1972, pp. 780-791.

6.12 D. J. Connor, "Techniques for Reducing the Visibility of Transmission Errors in Digitally Encoded Video Signals," *IEEE Transactions on Communications,* COM-21, No. 3, June, 1973, pp. 695-706.

6.13 T. Ishiguro, K. Iinuma, Y. Iijima, T. Koga and H. Kaneko, "NETEC System: Interframe Encoder for NTSC Color Television Signals," *Third International Conference on Digital Satellite Communications Record,* November, 1975.

6.14 H. Yasuda, F. Kanaya and H. Kawanishi, "1.544 Mbit/s Transmission of TV Signals by Interframe Coding System," *IEEE Transmissions on Communications,* COM-24, No. 10, October, 1976, pp. 1175-1180.

6.15 CCITT Recommendation T.4, *Standardization of Group 3 facsimile apparatus for document transmission.*

6.16 CCITT Recommendation T.6, *Facsimile coding schemes and coding control functions for Group 4 facsimile apparatus.*

Questions for Understanding

6.1 In Fig. 6.1.1 why is pel A so far from pel X? Would a closer pel ever be a better predictor? When?

6.2 Which statistical and perceptual redundancies are exploited by the Adaptive Predictive Interfield Coder of Sec. 6.2?

6.3 Which statistical and perceptual redundancies are exploited by the Interfield Hybrid Transform Coder of Sec. 6.3?

6.4 Which statistical and perceptual redundancies are exploited by the three coding modes of the CCIR 723 coding algorithm?

6.5 In the CCIR 723 coder, suppose a Macroblock has the lowest Activity level and a Buffer Fullness Factor 50. Also, the first four transmitted DCT coefficients have unquantized values -1000, 150, -875 and 690. What are the first four transmitted quantizer indices?

6.6 Which statistical and perceptual redundancies are exploited by the Conditional Replenishment DPCM Coders of Sec. 6.6?

6.7 Explain *Hysteresis* in buffer control of interframe coders. What does it accomplish?

6.8 Explain *Forced Updating* in interframe coders. What does it accomplish?

6.9 Which statistical redundancies are exploited by the bilevel image coders of Sec. 6.7?

7

Still Image Coding Standards - ISO JBIG and JPEG

7.1 JBIG Coding

The bi-level coding algorithm most recently standardized by ISO is the JBIG (Joint Bi-level Imaging Group) algorithm[7.1]. JBIG coding is more complex than G3/G4 coding, but offers two compensating advantages.

One is superior compression. On text or line-art images, JBIG achieves about 20 percent greater compression than G4, the most efficient of the G3/G4 algorithms. The JBIG compression advantage is much greater, generally ranging between a factor of two and ten, on halftone bi-level images rendering grayscale. The JBIG algorithm is adaptive and indirectly recognizes and adjusts to input images rendering grayscale. In addition, special features are built in to recognize and exploit any periodicities present in grayscale renderings.

The second advantage of JBIG coding is that, if desired, it can be parameterized for progressive coding.

7.1.1 Progressive Coding

Progressive, or synonymously, *multiresolution*, coding captures images as a compression of a low resolution rendition plus a sequence of *delta* files that each provide another level of resolution enhancement. In the case of JBIG, each delta file contains the information needed to double both the vertical and horizontal resolution.

Progressive coding is attractive in database applications where output devices of varying resolution capability (say PC screens, workstation screens,

---

This chapter contains excerpts taken with permission from AT&T Technical Journal[7.17].

and laser printers) are all to be served. Only the bottom-layer coding and as many delta files as are of use are transmitted and processsed. In contrast, if images are stored non-progressively, it is necessary to either store compressions at all resolutions or to store only compressions at the highest resolution but require output devices to first decode to high resolution and then map down to display resolution. The first alternative wastes storage capacity. The second wastes transmission capacity and processing power.

Another application for progressive coding is in image browsing over medium rate communication links. A low-resolution rendition of an image can be made available quickly and then followed by as much resolution enhancement as is desired. A user on seeing at low resolution that the image being developed is not that desired, can quickly interrupt its transmission and move on to the next. This advantage for progressive coding only accrues on medium rate links, roughly those with speeds between 9.6 kb/s and 64 kb/s when bi-level images are being retrieved. On slower links, no user would have the patience required for image browsing no matter what the form of presentation. On higher speed links, the image comes so fast relative to human reaction times that how it develops is immaterial.

A third potential application for progressive coding is on packet networks on which packets can or must be priority classified as droppable or non-droppable. The packets carrying the information for the final resolution doubling would be sent at low priority and if the network were to drop them, the only penalty would be a slightly less sharp image. No entire regions would be lost or destroyed.

The JBIG specification uses the symbol D to denote the number of resolution doublings (delta files) that are to be available. The parameter D is unrestricted and may be chosen as 0 when progression is of no benefit. Making this choice generally improves compression by about 5 percent, but surprisingly, can degrade compression by as much as 50% on grayscale renderings created by periodicity generating algorithms.

Common choices for D when progression is wanted are 4, 5, and 6. If the original is a 400 dpi scanning of an 8.5 by 11 inch page and D is chosen as 5, the bottom-layer image will be 107 by 138 pels in size. Normally such a small image would be blown up by pel replication (or perhaps something more sophisticated) before display. If instead, however, it is displayed dot for dot on something like a 1000 by 1000 pel display, it can serve nicely as an icon.

7.1.2 Functional Blocks

The JBIG algorithm is complex and describing it completely here is not possible. Such detail is best obtained from the specification itself[7.1]. As in the discussion of the G3/G4 algorithms, the goal here is only to convey the basics of the algorithm.

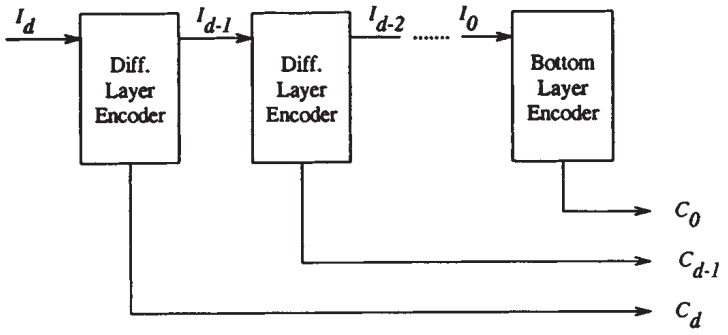


Fig. 7.1.1 Decomposition of a JBIG encoder.

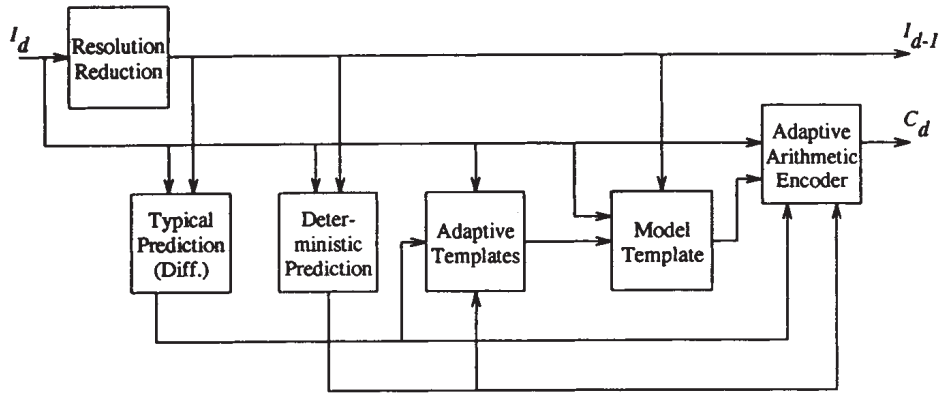


Fig. 7.1.2 Functional blocks in a JBIG differential-layer encoder.

Conceptually a JBIG encoder can be decomposed (see Fig. 7.1.1) into a chain of D differential layer encoders followed by a bottom-layer encoder. In Fig. 7.1.1, I_d denotes the image at layer d and C_d denotes its coding. A hardware implementation would in all likelihood time-share one physical differential-layer encoder, but for heuristic purposes the decomposition of Fig. 7.1.1 is helpful. Each differential layer encoder can be decomposed into the functional blocks shown in Fig. 7.1.2. The bottom-layer encoder has the somewhat simpler decomposition shown in Fig. 7.1.3.

7.1.3 Resolution Reduction

The resolution reduction block in a differential-layer encoder accepts the high-resolution image I_d and creates the low-resolution image I_{d-1} with, as nearly as possible, half as many rows and half as many columns.* A simple way of performing this resolution reduction would be by subsampling, that is, discarding every other row and every other column. Subsampling is simple, but the low-resolution images it creates are poorer in subjective quality than is necessitated by their diminished resolution alone. On images with line art, whole lines can be lost if they happen to lie on rows or columns being discarded. Grayscale renderings with periodicities frequently display aliasing artifacts caused by beats between the superpel frequency and the subsampling by two.

JBIG's resolution reduction algorithm works remarkably well for all image types — text, line art, and grayscale renderings. It is a table based algorithm. The low-resolution image is created pel by pel in the usual raster scan row order. The color of any given low-resolution pel is then dictated by the colors of nine particular high-resolution neighbors and three causally positioned low-resolution neighbors.

7.1.4 Arithmetic Coding

The remaining functional blocks in Fig. 7.1.2 and 7.1.3 implement the compression algorithm. The heart of this functionality in both the bottom-layer coder and the differential-layer coder is an adaptive arithmetic coder[7.2]. As we saw in Chapters 3 and 5, arithmetic coders are distinguished from other entropy coders such as Huffman coders and Ziv-Lempel coders in that, conceptually at least, they map a string of binary symbols to be coded into a real number x on the unit interval [0.0,1.0). In the JBIG application, the string of symbols to be coded is the pels of the image I_d presented in raster scan order. What is transmitted or stored instead of the image I_d is a binary representation of x.

---

* In general, there need not be even numbers of rows and columns in I_d.

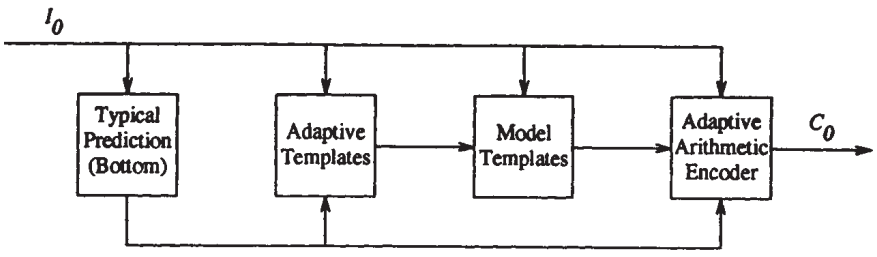


Fig. 7.1.3 Functional blocks in a JBIG bottom-layer encoder.

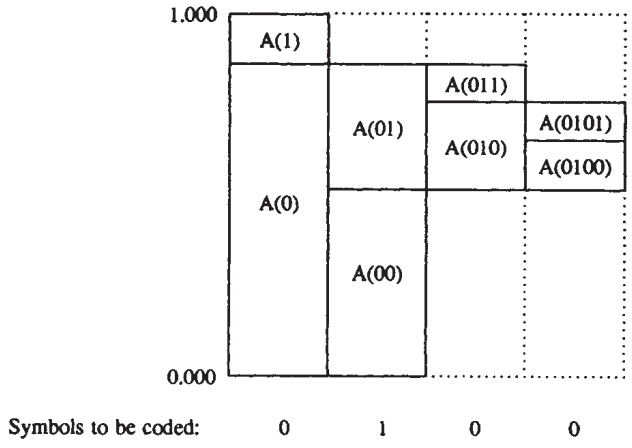


Fig. 7.1.4 Example of interval subdivision.

The particular number x to which the input sequence maps is determined by the recursive probability interval subdivision of the Elias coder[7.3] as described in Sec. 3.1.7 and Sec. 5.7.1e. Fig. 7.1.4 shows an example of such interval division through an initial sequence 0,1,0,0 to be coded.

The portion of the unit interval on which x is known to lie after coding each symbol is known as the *current coding interval*, and we designate its size by A. We define the current estimate of x to be the base of the current interval and denote this by C. For each new binary input the current coding interval is divided into two sub-intervals with sizes pA and $(1-p)A$, where p is the conditional probability of the Least Probable Symbol (LPS). The new current coding interval becomes that portion of the old coding interval associated with the new symbol value actually occurring. The fact that, most frequently, the symbol to be coded will be the Most Probable Symbol (MPS) and that by design this symbol codes into the bottom of the two sub-intervals means that most of the time the coding interval A is not reduced in size by as much as 1/2, C does not change and an input symbol is coded without generating any bits in the binary expansion of x. It is true that whenever an LPS does happen to occur, more than one output bit is generated. However, a central result of information theory is that on the average there is still less than one bit generated for each input symbol.

The Elias coder was conceived soon after Shannon's seminal paper gave birth to Information Theory in 1948[7.4]. It was of little practical use, however, until ways were found in the mid 1970's to perform all the necessary arithmetic with finite precision arithmetic and in a pipelined fashion so that a decoder could start its processing without having to wait for an encoder to finish its processing[7.5].

Sec. 5.7.1e described a finite precision arithmetic coding technique that implements the above interval subdivision algorithm. However, in JBIG the multiplication needed to compute pA is deemed too costly for inexpensive implementation. Instead, JBIG assumes that after normalization, A is usually not too far from some value $\bar{A}$, measured by averaging A over a wide variety of test documents. JBIG assumes a value of approximately 0.72 for $\bar{A}$.

With this assumption, the conditional probability tables that contain p can be replaced with values $p \times \bar{A}$, which we denote by $\bar{p}$. Thus, in the interval subdivision, LPS has size $\bar{p}$, and MPS has size $A - \bar{p}$, which avoids any multiplication in the algorithm.

One complication of this approach is that for some values of A we can have $\bar{p} > A - \bar{p}$, i.e., the nominal LPS size can exceed the nominal MPS size. When this happens, JBIG performs what is called a *conditional exchange*, which basically reverses the assignment of symbols to intervals so that the symbol with higher probability is always assigned to the largest subinterval. The complete operation of coding a symbol is as follows:

```
if (p̄ > A/2) exchange = true      //need conditional exchange
   else exchange = false
if( (!exchange && MPS) || (exchange && LPS) ){
   A = A − p̄                      //code bottom subinterval
}
else{                             //code top subinterval
   A = p̄
   C = C + A − p̄
}
```

Following the symbol coding, A and C are renormalized as described previously.

7.1.5 Model Templates

For each high-resolution pel to be coded, the model-templates block provides the arithmetic coder with an integer S called the context. For differential-layer coding S is determined by the colors of six adjacent previously transmitted pels in the causal high-resolution image, also by the colors of four particular pels in the already available low-resolution image, and also by the *spatial phase* of the pel being coded. The term spatial phase denotes which of the four possible positions the high-resolution pel has with respect to its corresponding low-resolution pel. Thus, there are $4 \times 2^6 \times 2^4 = 4096$ possible contexts for differential-layer coding. The six particular high-resolution pels and four particular low-resolution pels whose colors (along with spatial phase) define the context are known as the coding template or model template.

For bottom-layer coding, the coding template only includes 10 high-resolution pels for a total of 1024 contexts. There are no low-resolution pels to incorporate nor is there an analog of the spatial-phase concept.

The arithmetic coder generates for each context an estimate of the conditional probability $\overline{p}$ of the LPS given that context. Doing this well requires carefully balancing speed of adaptation against quality of estimation[7.6]. The greatest coding gain is achieved when this probability estimate is both accurate and close to 0. Thus, good templates must have good predictive value so that when the values of the pels in the template are known, the value of the pel to be coded is highly predictable.

The simplest implementation of Arithmetic Coding would utilize a fixed lookup table containing one $\overline{p}$ value for each context. However, JBIG allows for adaptation of the context probability values in order to better track the actual statistics of the image being coded. Basically, this adaptation works by decrementing $\overline{p}$ slightly whenever an MPS renormalization occurs, and incrementing $\overline{p}$ slightly whenever an LPS renormalization occurs. If any $\overline{p}$ value becomes too large, indicating an erroneous LPS symbol definition, then the LPS and MPS symbol definitions are interchanged. The adaptation process is

completely defined by the Probability Estimation State Machine shown in Fig. 7.1.5.

JBIG only allows for 112 possible $\overline{p}$ values. These are shown in the second column of Fig. 7.1.5 in hexadecimal, where the binary point is on the left. Thus, context statistics can be stored in two arrays Index[ ] and MPS[ ] of size 4096 containing, respectively, the indices of the $\overline{p}$ values and the most probable symbol values (0 or 1). At the start of coding, both arrays are initialized to zero. During coding of a symbol having context S, the LPS probability and most probable symbol value are $\overline{p}$[Index[S]] and MPS[S], respectively.

If coding causes a renormalization, then context statistics are adjusted according to the Fig. 7.1.5 state machine via the following algorithm:

```
                                //after renormalization...
I = Index[S]                    //save current p̄ index
if( symbol==MPS[S]){            //an MPS was coded
Index[S] = NextMPS[I]           //update p̄
}
else{                           //an LPS was coded
Index[S] = NextLPS[I]           //update p̄
if( Switch[I]==1 )              //an LPS was coded and p̄ is large
MPS[S] = ~MPS[S]                //interchange LPS and MPS definition
}
```

Note that the smallest $\overline{p}$ is 2^{-16}. Also, indices 0 to 8 are normally used only during startup of the statistical adaptation.

7.1.6 Adaptive Templates

The comparatively good compressions obtained by JBIG on dithered grayscale renderings is partially attributable to the adaptation within the arithmetic coder. The adaptive-templates (AT) block adds a factor of approximately two improvement on dithered grayscale renderings with periodicities. AT looks for periodicities in the image and, upon finding a strong indication for one, changes the template so that, as nearly as possible, a pel offset by this periodicity is incorporated into the new template. Such a pel of course has excellent predictive value.

Any changes of this sort to the template must only be made infrequently since a period of readaptation in the probability estimator ensues, and compression degrades while it occurs. An algorithm for determining the if, when, and how of any template rearrangement normally has substantial hysteresis so that template changes only occur when there is an extremely strong indication that a new template offers substantial improvement over the current one.

Index	$\overline{p}$	NextLPS	NextMPS	Switch		Index	$\overline{p}$	NextLPS	NextMPS	Switch
0	.5a1d	1	1	1		57	.01a4	55	58	0
1	.2586	14	2	0		58	.0160	56	59	0
2	.1114	16	3	0		59	.0125	57	60	0
3	.080b	18	4	0		60	.00f6	58	61	0
4	.03d8	20	5	0		61	.00cb	59	62	0
5	.01da	23	6	0		62	.00ab	61	63	0
6	.00e5	25	7	0		63	.008f	61	32	0
7	.006f	28	8	0		64	.5b12	65	65	1
8	.0036	30	9	0		65	.4d04	80	66	0
9	.001a	33	10	0		66	.412c	81	67	0
10	.000d	35	11	0		67	.37d8	82	68	0
11	.0006	9	12	0		68	.2fe8	83	69	0
12	.0003	10	13	0		69	.293c	84	70	0
13	.0001	12	13	0		70	.2379	86	71	0
14	.5a7f	15	15	1		71	.1edf	87	72	0
15	.3f25	36	16	0		72	.1aa9	87	73	0
16	.2cf2	38	17	0		73	.174e	72	74	0
17	.207c	39	18	0		74	.1424	72	75	0
18	.17b9	40	19	0		75	.119c	74	76	0
19	.1182	42	20	0		76	.0f6b	74	77	0
20	.0cef	43	21	0		77	.0d51	75	78	0
21	.09a1	45	22	0		78	.0bb6	77	79	0
22	.072f	46	23	0		79	.0a40	77	48	0
23	.055c	48	24	0		80	.5832	80	81	1
24	.0406	49	25	0		81	.4d1c	88	82	0
25	.0303	51	26	0		82	.438e	89	83	0
26	.0240	52	27	0		83	.3bdd	90	84	0
27	.01b1	54	28	0		84	.34ee	91	85	0
28	.0144	56	29	0		85	.2eae	92	86	0
29	.00f5	57	30	0		86	.299a	93	87	0
30	.00b7	59	31	0		87	.2516	86	71	0
31	.008a	60	32	0		88	.5570	88	89	1
32	.0068	62	33	0		89	.4ca9	95	90	0
33	.004e	63	34	0		90	.44d9	96	91	0
34	.003b	32	35	0		91	.3e22	97	92	0
35	.002c	33	9	0		92	.3824	99	93	0
36	.5ae1	37	37	1		93	.32b4	99	94	0
37	.484c	64	38	0		94	.2e17	93	86	0
38	.3a0d	65	39	0		95	.56a8	95	96	1
39	.2ef1	67	40	0		96	.4f46	101	97	0
40	.261f	68	41	0		97	.47e5	102	98	0
41	.1f33	69	42	0		98	.41cf	103	99	0
42	.19a8	70	43	0		99	.3c3d	104	100	0
43	.1518	72	44	0		100	.375e	99	93	0
44	.1177	73	45	0		101	.5231	105	102	0
45	.0e74	74	46	0		102	.4c0f	106	103	0
46	.0bfb	75	47	0		103	.4639	107	104	0
47	.09f8	77	48	0		104	.415e	103	99	0
48	.0861	78	49	0		105	.5627	105	106	1
49	.0706	79	50	0		106	.50e7	108	107	0
50	.05cd	48	51	0		107	.4b85	109	103	0
51	.04de	50	52	0		108	.5597	110	109	0
52	.040f	50	53	0		109	.504f	111	107	0
53	.0363	51	54	0		110	.5a10	110	111	1
54	.02d4	52	55	0		111	.5522	112	109	0
55	.025c	53	56	0		112	.59eb	112	111	1
56	.01f8	54	57	0						

Fig. 7.1.5 JBIG Probability Estimation State Machine. Probability values are hexadecimal with the binary point at the left.

7.1.7 Differential-Layer Typical Prediction

The differential-layer typical prediction (TP) block provides some coding gain, but its primary purpose is to speed implementations. Differential-layer TP looks for regions of solid color and when it finds that a given current high-resolution pel is in such a region, none of the processing normally done in the deterministic prediction, adaptive templates, model templates, or arithmetic coding blocks is needed. On text and line-art images differential-layer TP usually makes it possible to avoid coding about 95 percent of the pels. Grayscale renderings do not generally have the sought-after large regions of continuous color and do not allow processing savings like this.

The key idea behind differential-layer TP is that if a low resolution pel and all the pels in an eight-pel neighborhood of it are the same color, then it is extremely likely that all four high-resolution pels to be associated with it are that same color also. Unfortunately, it is not certain that this is so, but exceptions occur infrequently enough that it is efficient and reasonable to flag them. In particular, an encoder notes at the beginning of each high-resolution line pair whether or not a decoder would ever go wrong on that line pair if it always were to "typically expand" any low-resolution pels it found to be within a common-color eight-neighborhood into four high-resolution pels of that color. The failure or success of this strategy over the line pair is coded and sent to the decoder. Note that failure here is not that some low-resolution pel in the line pair is not within a common-color eight-neighborhood, but rather that some low-resolution pel to be associated with the line pair is within a common-color eight-neighborhood but will not have associated high-resolution pels of that color. Failure in this sense is extremely rare and on many images never occurs. When success over the line pair is coded, both the encoder and decoder skip over high-resolution pels associated with any low-resolution pels found to be within a common-color eight-neighborhood. If failure must be coded, the only penalty is that no skipping can be done for the line pair and everything must be coded.

7.1.8 Bottom-Layer Typical Prediction

Bottom-layer typical prediction, like differential-layer typical prediction, tries to exploit solid color regions of the image to save processing effort. However, the algorithms are quite different. The bottom-layer algorithm is a line-skipping algorithm. A given line is said to be "typical" and all its pels are declared *typical*, if it is identical to the line above it. Which lines are typical is again transmitted to the decoder. Both the encoder and decoder skip the coding of all pels in typical lines and generate them instead by line duplication.

It is not possible in bottom-layer coding to skip as large a percentage of pels as it is in differential-layer coding. On text and line-art images savings are generally about 40 percent.

7.1.9 Deterministic Prediction

When images are reduced in resolution by the JBIG resolution-reduction algorithm, it sometimes happens that the value of the particular high-resolution pel being coded is inferable from the pels already known to both the encoder and decoder, i.e., all the pels in the low-resolution image and those in the high-resolution image that are causally related in a raster sense to the current pel. When this occurs, the current pel is said to be deterministically predictable. The deterministic prediction (DP) block flags any such pels and inhibits their coding by the arithmetic coder. DP is a table driven algorithm. The values of particular surrounding pels in the low-resolution image and causal high-resolution image are used to index into a table to check for determinicity and, when it is present, obtain the deterministic prediction. DP provides about a 7 percent coding gain.

7.1.10 Compression Comparison

Fig. 7.1.6 shows compression performance on the eight standard CCITT test images and one additional image. The one additional image is a binary image rendering grayscale via halftoning. It is a picture of a Japanese woman holding flowers. The eight CCITT images are all sampled at 200 dpi (dot-per-inch) and contain 1728 by 2376 pels. The halftoned image contains 2304 by 2896 pels. Compressed-file byte counts are provided for coding with one-dimensional G3, two-dimensional G3 (k factor of 4), G4, non-progressive JBIG, and progressive JBIG with 4 delta layers.

Over the eight CCITT images, non-progressive JBIG coding has about a 22% coding advantage over G4, the most efficient of the G3/G4 algorithms. The progressive JBIG algorithm provides progressivity while at the same time still showing an average 15% coding gain over G4.

It's widely recognized that the G3/G4 algorithms are not suitable for coding bi-level images rendering grayscale via halftoning. This is readily apparent in the last row of Fig. 7.1.6 where the JBIG compression advantage is about a factor of five.

7.2 JPEG Still-Color-Image Coding

The need for an international standard for continuous-tone still image compression resulted, in 1986, in the formation of JPEG, which stands for Joint Photographic Experts Group. This group was chartered by ISO and the CCITT to develop a general-purpose standard suitable for as many applications as possible. After thorough evaluation and subjective testing of a number of proposed image-compression algorithms, the group agreed, in 1988, on a DCT-based technique. From 1988 to 1990, the JPEG committee refined several methods incorporating the DCT for lossy compression. A lossless method was also defined. The committee's work has been published in two parts: "Part 1: Requirements and guidelines"[7.7] describes the JPEG compression and decompression method. "Part 2: Compliance Testing"[7.8] describes tests to

Image	Bytes					
	Raw	G3D1	G3D2	G4	JBIGD0	JBIGD4
CCITT #1	513216	37423	25967	18103	14715	16771
CCITT #2	513216	34367	19656	10803	8545	8933
CCITT #3	513216	65034	40797	28706	21988	23710
CCITT #4	513216	108075	81815	69275	54356	58656
CCITT #5	513216	68317	44157	32222	25877	28086
CCITT #6	513216	51171	28245	16651	12589	13455
CCITT #7	513216	106420	81465	69282	56253	60770
CCITT #8	513216	62806	33025	19114	14278	15227
Halftone	834048	483265	572259	591628	131479	103267

Fig. 7.1.6. Compressed file sizes in bytes for various coding algorithms.

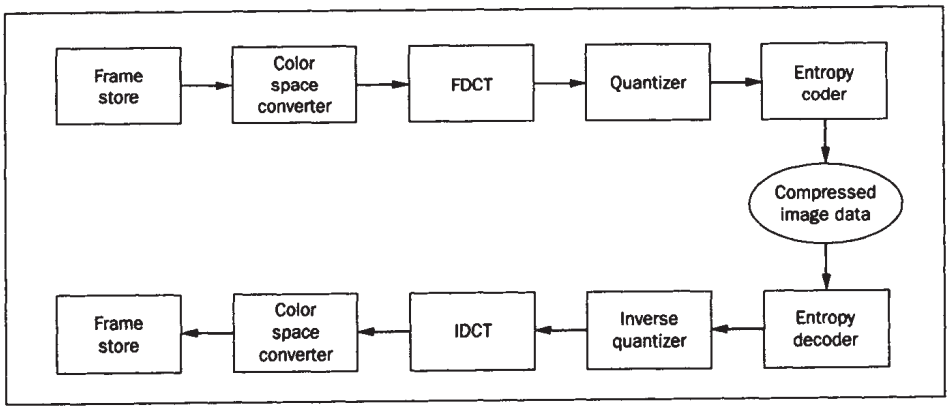


Fig. 7.2.1. Sequential DCT codec.

verify whether an encoder-decoder (codec) has implemented the JPEG algorithms correctly. Reference[7,9] describes the JPEG algorithms in detail.

To appreciate the need for a standard image compression algorithm, consider the storage/transmission requirements of an uncompressed image. A typical VGA digital color image has 640 pels × 480 lines. At three bytes per pel (one each for the red, green and blue components), such an image requires 921,600 bytes of storage space. To transmit the uncompressed image over a 64-kb/s ISDN channel takes about 1.9 minutes. The JPEG algorithms offer "excellent" quality for most images compressed to about 1.0 bit/pel. This 24:1 compression ratio reduces the required storage of the 640 × 480 color image to 38,400 bytes, and its transmission time to about 4.8 seconds. Applications for image compression may be found in desktop publishing, education, real estate, and security, to name a few.

In the next section, we give an overview of the JPEG algorithms. In subsequent sections, we present some operating parameters and definitions, and describe each of the JPEG operating modes in more detail.

7.2.1 Overview of the JPEG Algorithms

The JPEG committee could not satisfy the requirements of every still-image compression application with one algorithm. As a result, the committee defined four different modes of operation:

- *Sequential DCT-based* — Fig. 7.2.1 presents a simplified diagram of a sequential DCT codec. In this mode, 8 × 8 blocks of the input image are formatted for compression by scanning the image left to right and top to bottom. A block consists of 64 samples of one color component that make up the image. Each block of 64 samples is transformed to a block of 64 coefficients by the forward discrete cosine transform (DCT). The DCT concentrates most of the energy of the component samples' block into a few coefficients, usually in the top-left corner of the DCT block. The coefficient in the immediate top-left corner is called the DC coefficient because it is proportional to the average intensity of the block of spatial domain samples. AC coefficients corresponding to increasingly higher frequencies of the sample block progress away from the DC coefficient. The coefficients are then quantized and entropy-coded. A subset of the Sequential mode is the *Baseline Mode*, which is required to be present in all JPEG implementations.

- *Progressive DCT-based* — This mode offers a means of producing a quick "rough" decoded image when the medium separating the encoder and decoder has a low bandwidth. The method is similar to the sequential DCT-based algorithm, but the quantized coefficients are partially encoded in multiple scans.

- *Lossless* — In this mode, the decoder receives an exact reproduction of the digital input image. The differences between input samples and predicted values, where the predicted values are combinations of one to three neighboring samples, are entropy-coded.
- *Hierarchical* — This mode, also known as Pyramid coding, is used to encode an input image as a sequence of increasingly higher-resolution frames in exactly the same way as in JBIG. The first frame is a reduced resolution version of the original. Subsequent frames are coded as higher-resolution differential frames.

The color space conversion process in Fig. 7.2.1 is not a part of the standard. In fact, JPEG is color-space-independent. As a first step in the compression process, many image-compression schemes take advantage of the human visual system's low sensitivity to high-frequency chrominance information by reducing the chrominance resolution. Many images (usually RGB) are typically converted to a luminance-chrominance representation, e.g., YCbCr, before this processing takes place. Input pel values having precision n bits are unsigned integers in the range 0 to $2^n - 1$.

Either Huffman or arithmetic coding techniques can be used for entropy coding in any of the JPEG modes of operation (except the *baseline*, where Huffman coding is mandatory). There are no standard JPEG Huffman codes. The encoder must transmit its Huffman code tables (up to 4 AC, and 4 DC) according to a prescribed format. As described in Chapter 3, a Huffman coder compresses a series of input symbols by assigning short code words to frequently occurring symbols and long code words to improbable symbols.[7.10,7.11] The output of an arithmetic coder, on the other hand, is a single real number for each color component image. Unlike a Huffman coder, an arithmetic coder does not require an integral number of bits to represent an input symbol. As a result, arithmetic coders are usually more efficient than Huffman coders.[7.12,7.13] For the JPEG test images, Huffman coding (using fixed tables) resulted in compressed data requiring, on average, 13.2 percent more storage than arithmetic coding.

7.2.2 JPEG Operating Parameters and Definitions

A number of parameters related to the source image and the coding process may be customized to meet the user's needs. In this section, we discuss some of the important variable parameters and their allowable ranges. Also, as an aid to the algorithm descriptions in the following sections, we define some JPEG terms and present the hierarchical structure of the compressed data.

Parameters

An image to be encoded using any JPEG mode may have from 1 to 65,535 lines and from 1 to 65,535 pels per line. Each pel may have from 1 to 255 color components (at most 4 components are allowed for progressive mode). The

operating mode determines the allowable pel precision of the color component. For the DCT modes, either 8 or 12 bits of (unsigned) pel precision are supported (only 8 bit precision is allowed for *baseline*). Prior to coding, unsigned pel values, having precision n bits, are shifted downward by subtracting 2^{n-1} to give a signed range of -2^{n-1} to $2^{n-1}-1$. The lossless mode pel precision may range from 2 to 16 bits. If a DCT operating mode has been selected, the coefficient quantizer precision must also be defined. For 8-bit component precision, the quantizer precision is fixed at 8 bits. Twelve-bit components require either 8- or 16-bit quantizer precision.

Data interleaving

To reduce the processing delay and/or buffer requirements, JPEG can interleave up to four color components in a single scan (for progressive mode, only the DC scan may have interleaved components). A data structure called the *minimum-coded unit* (MCU) has been defined to support this interleaving. An MCU consists of one or more data units, where a data unit is a component sample for the lossless mode, and an 8×8 block of component samples for the DCT modes. If a scan contains only one component, then its MCU is equal to one data unit. For multiple component scans, the MCU for the scan consists of interleaved data units. The maximum number of data units per MCU is 10. As an interleaving example, consider a CCIR-601 digital image in which the chrominance components are subsampled 2:1 horizontally. For a DCT coder, a CCIR-601 MCU could consist of two Y blocks, followed by a Cb block and a Cr block.

Marker codes

JPEG has defined a number of two-byte marker codes to delineate the various sections of a compressed data stream. All marker codes begin with a byte-aligned hexadecimal ''FF'' byte, making it easy to scan and extract parts of the compressed data without actually decoding it. It is also possible to create by chance a byte-aligned hexadecimal ''FF'' byte within the normal entropy-coded data. Thus, the encoder must detect this situation and follow a byte aligned ''FF'' byte with a zero byte in order to avoid ambiguity. When the decoder encounters the byte aligned hexadecimal ''FF00'' combination, the zero byte must be removed.

Compressed-image data structure

At the top level of the compressed data hierarchy is the *image* (see Fig. 7.2.2). A nonhierarchical mode image consists of a *frame* surrounded by ''Start'' and ''End of Image'' marker codes. There will be multiple *frames* in a hierarchical mode image. Within a frame, a start of frame (SOF) marker identifies the coding mode to be used. The SOF marker is followed by a number of parameters (see Reference 7.7), and then by one or more *scans*. Each scan

begins with a header identifying the components to be contained within the scan, frequency coefficients contained in the scan for progressive mode, and more parameters. The scan header is followed by an entropy-coded segment (ECS). An option exists to break the ECS into chunks of MCUs called *restart intervals*. The restart interval structure is useful for identifying select portions of a scan, and for recovery from limited corruption of the entropy-coded data. Quantization and entropy-coding tables may either be included with the compressed image data or communicated separately.

7.2.3 Baseline Sequential DCT

The sequential DCT mode offers very good compression ratios, while maintaining excellent image quality. A subset of the sequential DCT capabilities has been identified by JPEG as a *Baseline System* (8-bit pels, Huffman coding, 8-bit quantizer precision, up to 2 AC and 2 DC Huffman Tables). All DCT-based JPEG implementations are required to include baseline capability. This requirement should help to ensure interoperability between codecs from different vendors.

The following subsections describe the processing steps for a baseline coder. A decoder is formed by reversing the coder steps.

DCT and quantization

All JPEG DCT-based coders begin the coding process by partitioning the input image into non-overlapping 8 × 8 blocks of component samples. After level-shifting the 8-bit samples so that they range from -128 to +127, the blocks are transformed to the frequency domain using the DCT[7.14,7.15] as defined in Chapter 5. The equations for the forward and inverse orthonormal discrete cosine transforms* are given by:

$$DCT: \quad c_{uv} = \frac{1}{4} K_u K_v \sum_{x=0}^{7} \sum_{y=0}^{7} b_{xy} \cos\frac{\pi u(2x+1)}{16} \cos\frac{\pi v(2y+1)}{16} \tag{7.1}$$

* These equations are exactly equivalent to the matrix representations defined in Chapters 3 and 5.

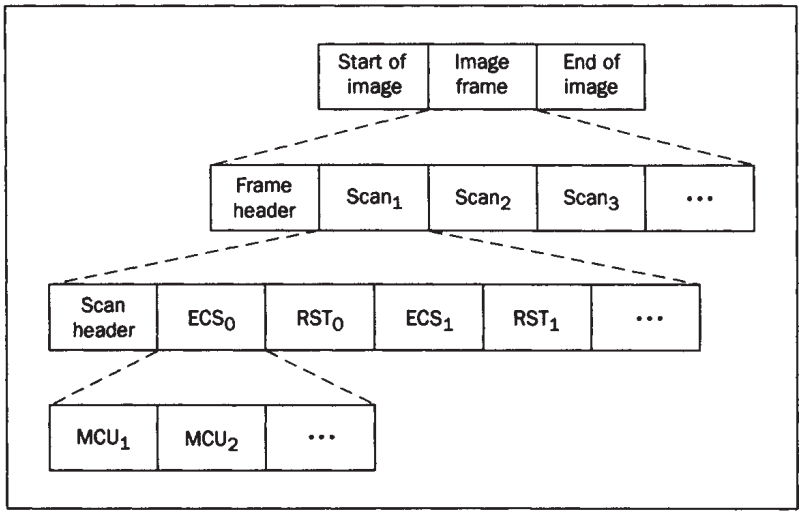


Fig. 7.2.2. Compressed-image data structure.

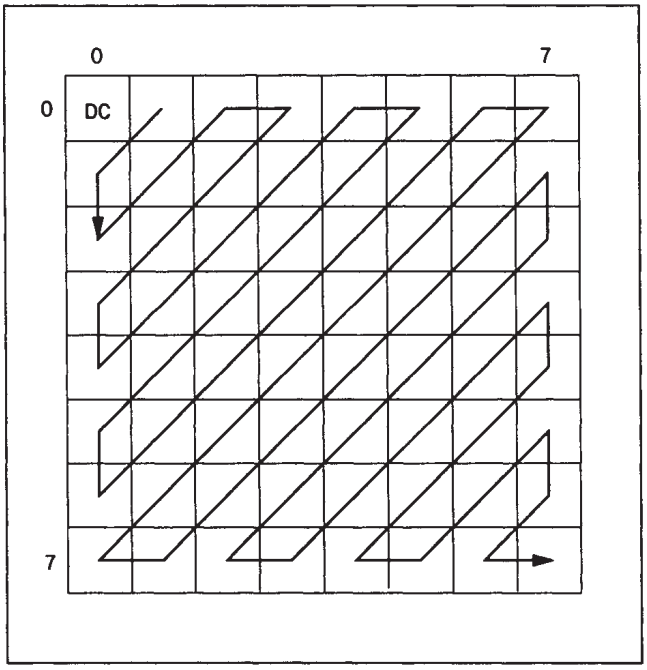


Fig. 7.2.3. Zigzag scan.

$$IDCT: \quad b_{xy} = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} K_u K_v c_{uv} \cos\frac{\pi u(2x+1)}{16}\cos\frac{\pi v(2y+1)}{16} \tag{7.2}$$

where

$$K_w = \frac{1}{\sqrt{2}} \text{ for } w = 0;\ K_w = 1 \textit{ otherwise.}$$

The next step in the process, quantization, is the key to most of the JPEG compression. For each picture, a 64-element quantization matrix $Q[u,v]$ is defined, where each element corresponds to a coefficient in the DCT block. $Q[u,v]$ is used to reduce the amplitude of the coefficients, and thus to increase the number of zero-value coefficients after quantization. Although there are no standard JPEG quantization matrices, JPEG includes a set that gives good results for CCIR-601 type images. The encoder must transmit its quantization matrices (up to 4) according to a prescribed format. The quantization and dequantization is performed according to Eq. (7.3) and Eq. (7.4), respectively.

$$FQ_{uv} = round\left(\frac{c_{uv}}{Q[u,v]}\right) \tag{7.3}$$

$$\tilde{c}_{uv} = FQ_{uv} \times Q[u,v] \tag{7.4}$$

A carefully designed quantization matrix will produce high compression ratios while introducing negligible "visible" distortion.[7.16] Many JPEG implementations control the compression ratio (and output image quality) by using a *q-factor*, which is usually just a scale factor applied to the quantization matrices.

DC coefficient entropy coding

Greater compression efficiency can be obtained if a simple predictive method is used to entropy-code the DC coefficient separately from the AC coefficients. Recall that the DC coefficient corresponds to the average intensity of the component block. Since adjacent blocks will probably have similar average intensities., it is advantageous to encode the *differences* between the DC coefficients of adjacent blocks rather than their values. Each differential DC value is coded using a variable-length code (VLC) followed by a variable-length integer (VLI). The VLC, which is Huffman coded, indicates the size in bits of the VLI. The VLI simply gives the differential DC value.

Zigzag scan and AC coefficient entropy coding

After quantization, the coefficient blocks usually contain many zero-value AC coefficients. If the coefficients are reordered, using the zig-zag scan illustrated in Fig. 7.2.3, there will be a tendency to have long runs of zeroes. Only the nonzero AC coefficients are entropy-coded. As in the DC coefficient coding, a VLC-VLI pair results from the coding of an AC coefficient. However, in this case the AC VLC corresponds to two pieces of information: the number of zeroes (run) since the last nonzero coefficient, and the size of the VLI following the VLC. Two additional symbols are also assigned Huffman codes. The End-of-Block (EOB) symbol indicates that all remaining coefficients in the zigzag scan are zero. If the last coefficient of the zigzag scan is nonzero, the EOB is not sent. The zero-run-length (ZRL) symbol indicate a run of 16 zeros, which is needed since VLC can only code zero runs up to length 15.

7.2.4 Sequential DCT with Arithmetic Coding

As stated previously, entropy coding of DCT coefficients can use either Huffman or Arithmetic coding, with Arithmetic Coding giving somewhat better coding efficiency. As with Huffman Coding, Arithmetic Coding codes the DC difference first, followed by the AC coefficients in the zigzag order of Fig. 7.2.3. Coding typically ends after the last nonzero coefficient of the block.

The first step is to convert the DC difference (*DIFF*) into a string of bits amenable to Arithmetic Coding. If $DIFF = 0$, the first and only bit (except for EOB) is zero. Otherwise, the first bit is one, and coding continues by defining the decremented absolute value $Sz = |DIFF| - 1$. The second bit is the sign bit. The next set of bits corresponds to the Huffman Coding VLC bits, and is related to the size in bits of *Sz*. The VLC bit pattern consists of $0 \le P \le 15$ ones followed by a zero, and indicates that $2^{P-1} < |DIFF| \le 2^P$. If $P \ge 2$, the VLC bits are followed by the VLI, which consists of the $P-1$ least significant bits of *Sz*. Finally, comes the EOB bit, which is one if there are no more AC coefficients to code, and is zero otherwise.

This DC bit string is then Arithmetic Coded using a 50 entry context table along with the JBIG probability table of Fig. 7.1.5. The first bit is coded using one of five contexts depending on an integer parameter $0 \le D_a \le 4$, which in turn depends on the magnitude and sign of the DC difference in the previous block. Similarly, for the sign bit. The context of the first VLC bit depends not only on D_a, but also on the sign bit, for a total of 10 possible contexts. The remaining VLC bits each have their own context, independent of anything else. The VLI bits are all coded with the same context that depends only on *P*. The DC EOB bit has its own context.

Each AC coefficient value (*ACVAL*) is then coded in a very similar manner. First it is converted to a string of bits in the same way as the DC difference, with two important exceptions. If $ACVAL = 0$, no EOB bit is produced since the decoder knows by convention that another AC coefficient will be sent. Also, if

all 64 coefficients of the block are sent, no EOB bit is produced after the last coefficient.

This AC bit string is then Arithmetic Coded using a 244 entry context table along with the JBIG probability table of Fig. 7.1.5. The first bit is coded using one of 63 contexts depending on the position $1 \le K \le 63$ of the coefficient in the zigzag sequence. The sign bit is almost completely random. Therefore, it is coded without context using a fixed probability ($MPS = 0$, $\bar{p} =$ .5A1D hexidecimal). The first two VLC bits use the same context, which again depends only on K, thus requiring another 63 contexts. The remaining VLC bits (up to 14) each have two contexts that depend on whether K exceeds a predefined threshold K_x, thus requiring another 28 contexts. The VLI bits (up to 14) are all coded with the same context that depends only on P and whether $K > K_x$. This requires another 28 contexts. Finally, the EOB bit context depends only on K, thus requiring another 62 contexts ($K = 63$ has no EOB bit).

7.2.5 Progressive DCT

A progressive DCT mode has been defined by JPEG to satisfy the need for a fast decoded picture when a low-bandwidth medium separates an encoder and decoder. By partially encoding the quantized DCT coefficients in multiple scans, the decoded image quality builds progressively from a coarse level to the quality attainable with the quantization matrices. Either spectral selection, successive approximation, or a combination of the two is used to code the quantized coefficients.

Spectral selection: In this method, the quantized DCT coefficients of a block are first partitioned into non-overlapping bands along the zigzag block scan. The bands are then coded in separate component scans. Before an AC coefficient band of a component may be coded, its DC coefficient must be coded. DC coefficients from as many as four components may be interleaved in a single scan. Interleaving is not permitted for AC bands.

Successive approximation: With this method, the precision of the coefficients is successively increased during multiple scans. Following a scan for a specified number of most significant bits of the quantized coefficients, subsequent scans increase the precision in increments of one bit until the least significant bits have been coded.

7.2.6 Lossless Mode

The lossless mode was defined for applications in which output pels from a decoder must be identical to the input pels to the encoder. The compression ratios achievable with the lossless mode, typically around 2:1, are much smaller than those afforded by the lossy modes. It does not use the DCT. Instead it uses DPCM in a way similar to that used to code the DC coefficients in the DCT-based modes, except that the predictor is selectable from one of seven choices, as shown in Fig. 7.2.4. Samples a, b, and c in the table correspond to neighbors

Selection value	Prediction
0	No prediction
1	a
2	b
3	c
4	a + b - c
5	a + ((b - c)/2)
6	b + ((a - c)/2)
7	(a + b)/2

Fig. 7.2.4. Lossless Mode Predictors.

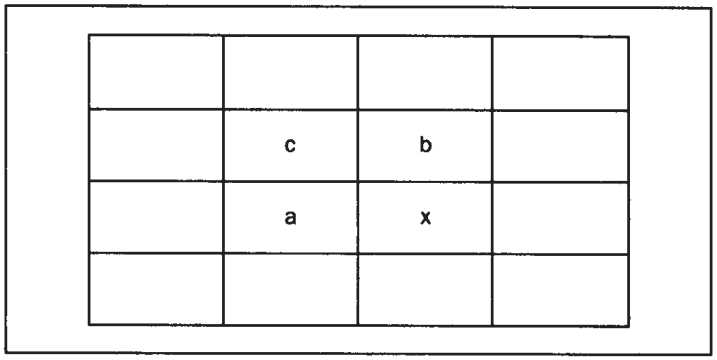


Fig. 7.2.5. Prediction neighborhood.

of the sample x to be predicted. Fig. 7.2.5 illustrates the prediction neighborhood. Entries 1 to 3 in Fig. 7.2.4 are used for one-dimensional predictive coding, and 4 through 7 form two-dimensional predictors. Entry 0 identifies differential coding for the hierarchical mode. As in the DC coefficient entropy coding described earlier, differences between the actual and predicted values are entropy-coded.

7.2.7 Hierarchical Mode

In the hierarchical mode, an image is coded as a succession of increasingly higher-resolution frames. Also known as *pyramidal* coding, this approach offers a higher quality alternative to the previously described methods for achieving progression. However, it is more expensive to implement. It also allows decoders with different resolution capabilities to use the same compressed data stream.

The first coded frame is created by reducing the resolution of the input image by a power of two in one or both dimensions, and then processing the lower resolution image using one of the lossy or lossless techniques of the other operating modes. Subsequent frames are formed by upsampling the decoded image by a factor of two in the dimension(s) having reduced resolution, subtracting the upsampled image from the input image at the same resolution, and coding the difference. This process continues until the decoded image has the same resolution as the full-resolution input image. After that, one or more full-resolution difference images may be coded. A hierarchical decoder may abort the decoding process after it has decoded a frame that provides the desired resolution.

Any coding methods described in the other three modes of operation may be used to code the hierarchical mode frames, with the following restrictions:

- If a lossy method is chosen, all but the last frame must be coded using that method. A lossless method optionally may be used to code the last frame.
- If a lossless method is chosen, all frames must be coded with that method.
- The same entropy-coding technique (Huffman or arithmetic) must be used for all frames.

The hierarchical coding/decoding process is not symmetrical. Indeed, a hierarchical coder must also include the greater part of a decoder. However, a hierarchical decoder is more complex than a nonhierarchical decoder because it must provide a way to upsample and add. This increased complexity may be justified, given the flexibility afforded in matching the decoder to the application. This type of codec is well suited for *one-to-many* applications, as in a number of decoders (possibly having different resolution capabilities) accessing a database of images precoded by a hierarchical coder.

References

7.1 ISO Draft International Standard 11544, *Coded representation of picture and audio information — Progressive bi-level image compression*, 1992.

7.2 T. C. Bell, J. G. Cleary, and I. H. Witten, *Text Compression*, Prentice Hall, 1990.

7.3 N. Abramson, *Information Theory and Coding*, pp. 61-62, McGraw-Hill, 1963.

7.4 C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, pp. 379-423 and 623-656, 1948.

7.5 F. Rubin, "Arithmetic stream coding using fixed precision registers," *IEEE Trans. Inf. Theory*, vol. IT-25, no. 6, pp. 672-675, November 1979.

7.6 C. Chamzas and D. Duttweiler, "Probability estimation in arithmetic and adaptive-Huffman entropy coders," submitted to *IEEE Transactions on Acoustics, Speech, and Signal Processing*.

7.7 *Digital Compression and Coding of Continuous-Tone Still Images, Part 1: Requirements and Guidelines*, ISO/IEC IS 10918-1, 1991.

7.8 *Digital Compression and Coding of Continuous-Tone Still Images, Part 2: Compliance Testing*, ISO/IEC IS 10918-2, 1991.

7.9 W. B. Pennebaker and J. L. Mitchell, *JPEG - Still Image Data Compression Standard*, Van Nostrand Reinhold, new York, 1993.

7.10 D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proc. IRE*, No. 40, September 1952, pp. 1098-1101.

7.11 J. Amsterdam, "Data Compression with Huffman Coding," *BYTE*, Vol. 11, No. 5, May 1986, pp. 99-108.

7.12 G. G. Langdon, Jr., "An Introduction to Arithmetic Coding," *IBM J. Res. Develop.*, Vol. 28, No. 2, March 1984, pp . 135-149.

7.13 I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic Coding for Data Compression," *Commun. ACM*, Vol. 30, No. 6, June 1987, pp. 520-540.

7.14 N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, Vol. C-23, No. 1, January 1974, pp. 90-93.

7.15 R. J. Clarke, *Transform Coding of Images*, Academic Press, Orlando, Florida, 1985.

7.16 H. Lohscheller, "A subjectively adapted image communication system," *IEEE Transactions on Communications*, Vol. COM-32, December 1984, pp. 1316-1322.

7.17 R. Aravind, *et al*, "Image and Video Coding Standards," *AT&T Technical J.*, Vol. 72, No. 1, January 1993, pp. 67-89.

Questions for Understanding

7.1 Why is the G3/G4 coding algorithm inefficient on dithered halftone bilevel images?

7.2 What advantages does progressive coding offer?

7.3 What is $\overline{p}$ in JBIG coding? Why does JBIG use $\overline{p}$ instead of p?

7.4 Why is *conditional exchange* needed in JBIG?

7.5 What is *context* in JBIG coding? How is it computed? How many possible context values are there for layered and non layered coding?

7.6 For a context value of 1346, what is the starting value of $\bar{p}$?

7.7 Suppose in the example above the next three pels have context values 1580, 1346, 1346 and all are LPS. What $\bar{p}$ values are used in coding these three pels?

7.8 Explain how Adaptive Templates work in coding dithered gray scale images.

7.9 Explain how Typical Prediction works in JBIG.

7.10 What are the four operating modes of JPEG coding?

7.11 How are Huffman codes determined in JPEG?

7.12 Of what use is data interleaving in JPEG? What is an MCU?

7.13 Why are Marker Codes used? How are they inserted into the compressed bit stream?

7.14 Briefly, describe the JPEG Baseline coding algorithm.

7.15 Write the bit strings used by JPEG Arithmetic Coding to represent DCDIFF or AC values between -20 and 20.

7.16 Write the bit string input to a JPEG Arithmetic Coder when the DCDIFF and first five AC values are -10, 8, 0, 0, -5, 20.

7.17 Explain the Hierarchical JPEG mode.

8

CCITT H.261 (P*64) Videoconferencing Coding Standards

8.1 Video Coding at Sub-Primary Rates

In most countries there is only limited availability of bit-rates below the primary rates, although the situation is changing rapidly. Two bit-rates that have been established for Integrated Services Digital Networks (ISDN) are of interest for image transmission. They are the so called B-channel of 64 kbits/s and the H0-channel of 384 kbits/s.

Many countries are offering a so called *Basic Rate* ISDN, which consists of two B-channels plus a signalling channel (2B+D). Basic Rate ISDN enables video coding at 112 kbits/s and speech coding at 16 kbits/s. H0 ISDN allows video coding at 320 kbits/s and audio at 64 kbits/s.

Standard international video coding algorithms have been established for ISDN bit-rates. In this section, we will outline some of the methodology that is used for video codecs operating at sub-primary bit-rates.

Coding of motion video at these rates usually requires not only conditional replenishment, but also fairly high data compression down to rates well below 1 bit per moving-area pel. The most straightforward way of achieving this is to code two-dimensional blocks of data using a transform such as the Discrete Cosine Transform (DCT) followed by Variable Length Codes (VLCs) such as Huffman Codes. Additional efficiency can be achieved by using motion compensation to subtract predictions from the moving-area pel values prior to transformation, as well as adaptive quantization and coding to maximize the received image quality for the channel capacity at hand.

This chapter contains excerpts taken with permission from AT&T Technical Journal[7.17].

8.2 From JPEG to H.261 (P*64)

From an algorithmic point of view, the extension from JPEG intraframe DCT coding to H.261[8.1] motion-compensated DCT video coding is a rather natural one. However, historically H.261 was developed long before JPEG. In December 1984, CCITT* Study Group XV (Transmission Systems and Equipment) established a "Specialists Group on Coding for Visual Telephony." The development of this video transmission standard for low-bit-rate ISDN services has gone through several stages. At the beginning, the design target was for $m\times384$ kbits channels, where m was between 1 and 5. Later, $n\times64$ kbits transmission rates (n from 1 to 5) were considered. However, by late 1989, the final CCITT recommendations were made for a $p\times64$ kbits/s video codec, where p is between 1 and 30, whence came the name P*64.

In fact, P*64 (or H series) is a group of audiovisual teleservices standards (or recommendations) consisting of H.221 frame structure; H.230 frame synchronous control; H.242 communication between audiovisual terminals; H.320 systems and terminal equipment; and H.261 video codec. Audio codecs at several bit rates have also been specified by other ITU-T recommendations, such as G.711, G.722, G.728. In this chapter, we concentrate on the H.261 video codec system.

Both JPEG baseline and H.261 codecs use DCT and VLC techniques. The major difference between JPEG and H.261 is the way each handles motion information. In JPEG, incoming picture frames are processed independently using intraframe DCT, while H.261 uses a block-based, motion-compensated scheme as described in Section 5.2.3g. That is, the picture data in the previous frame can be used to predict the image blocks in the current frame, as shown in Fig. 8.1. As a result, only differences, typically of small magnitude, between the displaced previous block and the current block have to be transmitted.

There are several interesting characteristics or design considerations in H.261.

- First, it defines essentially only the *decoder*. However, the *encoder*, which is not completely and explicitly specified by the standard, is expected to be compatible with the well-defined decoder.
- Second, because H.261 is designed for real-time communications, it uses only the closest previous frame for motion picture sequence coding to reduce the encoding delay.
- Third, it tries to balance the hardware complexities between the encoder and the decoder, because they are both needed for real-time videophone

* The CCITT has since renamed itself the ITU-T.

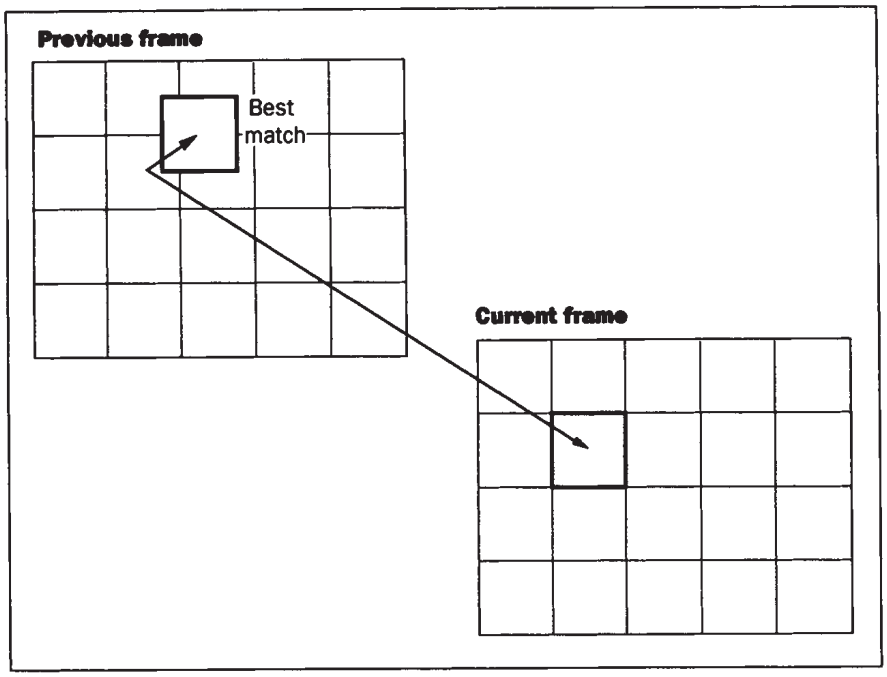


Fig. 8.1. Block motion compensation.

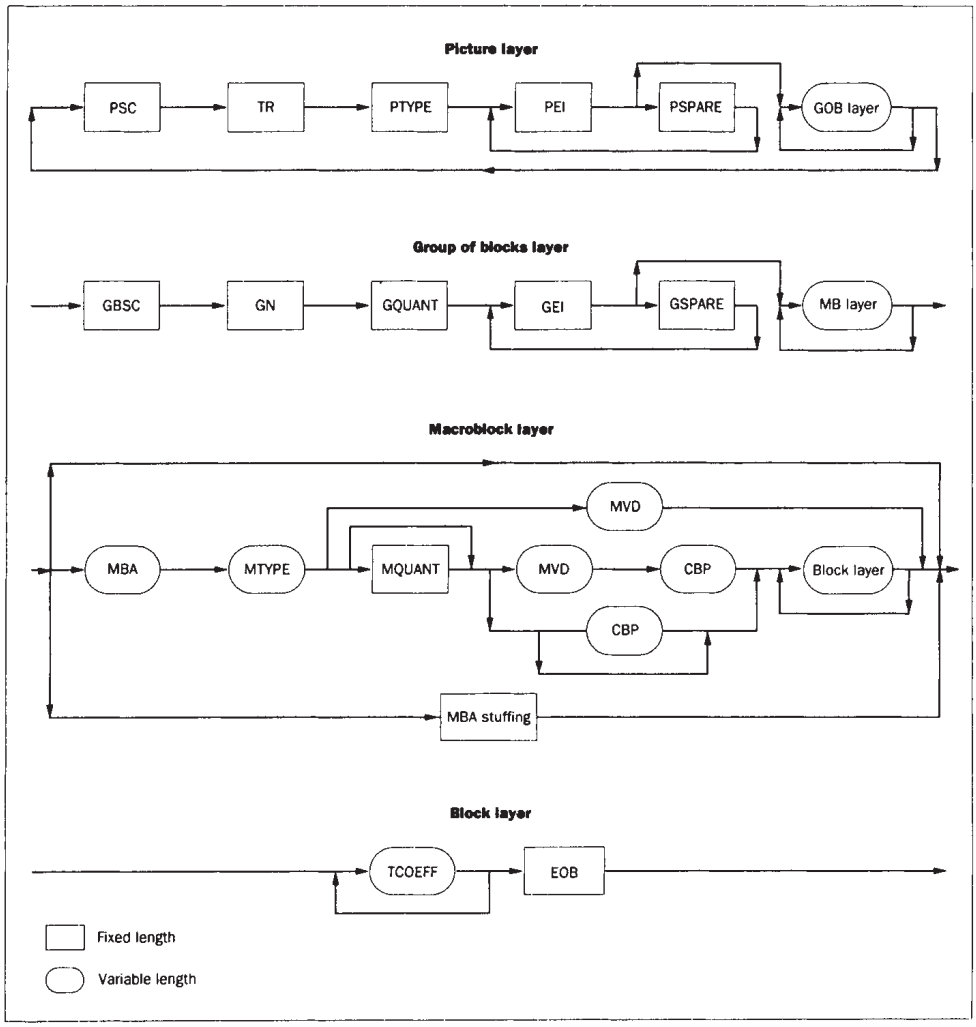


Fig. 8.2. Syntax diagram of the video multiplex coder.

applications. Other coding schemes, such as vector quantization (VQ), may have a rather simple decoder, but a very complex encoder.

- Fourth, H.261 is a compromise between coding performance, real-time requirements, implementation complexity, and system robustness. Motion-compensated DCT coding is a mature algorithm, and after years of study, quite general and robust in that it can handle various types of pictures.
- Fifth, the final coding structures and parameters are tuned more toward low-bit-rate applications. This choice is logical, because selection of the coding structure and coding parameters is more critical to codec performance at very low bit rates. At higher bit rates, less-than-optimal parameters values do not affect codec performance very much.

8.3 Decoder Structures and Components

H.261 specifies a set of protocols that every compressed video bit stream has to follow, and a set of operations that every standard compatible decoder must be able to perform. The actual hardware codec implementation and the encoder structure can vary drastically from one design to another. We will first explain briefly the data structure in an H.261 video bit stream and then the functional elements in an H.261 decoder.

The compressed H.261 video bit stream[8.1] contains several layers (see Fig. 8.2). They are *picture* layer, group of blocks (*GOB*) layer, Macroblock (*MB*) layer, and *block* layer. The higher layer consists of its own header followed by a number of lower layers.

Only two picture formats — common intermediate format (CIF) and quarter-CIF (QCIF) — are allowed. CIF pictures are made of three components: luminance Y and color differences C_B and C_R, as defined in CCIR Recommendation 601. The CIF picture size for Y is 352 pels per line by 288 lines per frame. The two-color difference signals are subsampled to 176 pels per line and 144 lines per frame. Fig. 8.3 shows the sampling pattern of Y, C_B, and C_R. The image aspect ratio is 4(horizontal):3(vertical), and the picture rate is 29.97 non-interlaced frames per second. Fig. 8.4 shows specifications for CIF and QCIF. All standard codecs must be able to operate with QCIF; CIF is optional.

A picture frame is partitioned into 8 line by 8 pel image blocks. A so-called Macroblock (MB) is made of 4 Y blocks, one C_B block, and one C_R block at the same location, as shown in Fig. 8.5a. Fig. 8.5b contains 33 MBs grouped into a GOB. Therefore, one CIF frame contains 12 GOBs and one QCIF frame 3 GOBs, as shown in Fig. 8.5c.

Picture Layer

In a compressed video bit stream, we start with the picture layer. Its header contains:

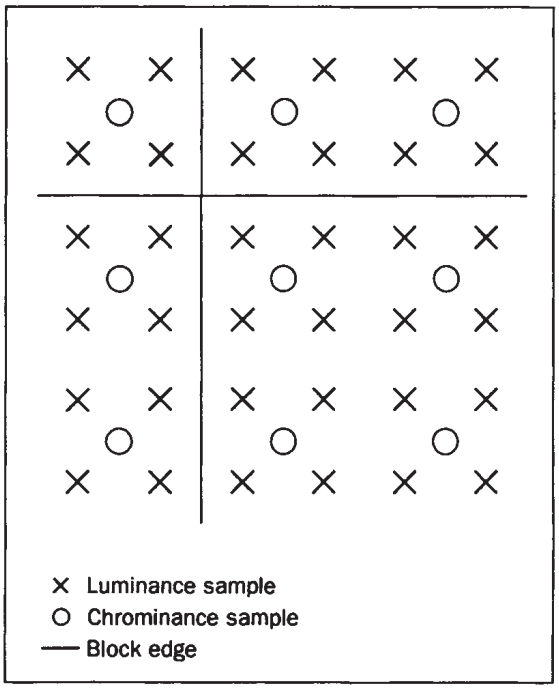


Fig. 8.3. Positioning of luminance and chrominance samples.

	CIF	QCIF
Luminance Pels per Line	352	176
Chrominance Pels per Line	176	88
Luminance Lines per Field	288	144
Chrominance Lines per Field	144	72
Fields per Second	29.97	29.97
Interlace	1:1	1:1
Color Components	Y $C_B C_R$	Y $C_B C_R$
Luminance Range	16 - 235	16 - 235
Chrominance Range	16 - 240	16 - 240
Zero Color Difference Level	128	128
Bits per Pel	8	8

Fig. 8.4. Internationally compatible videoconferencing video formats.

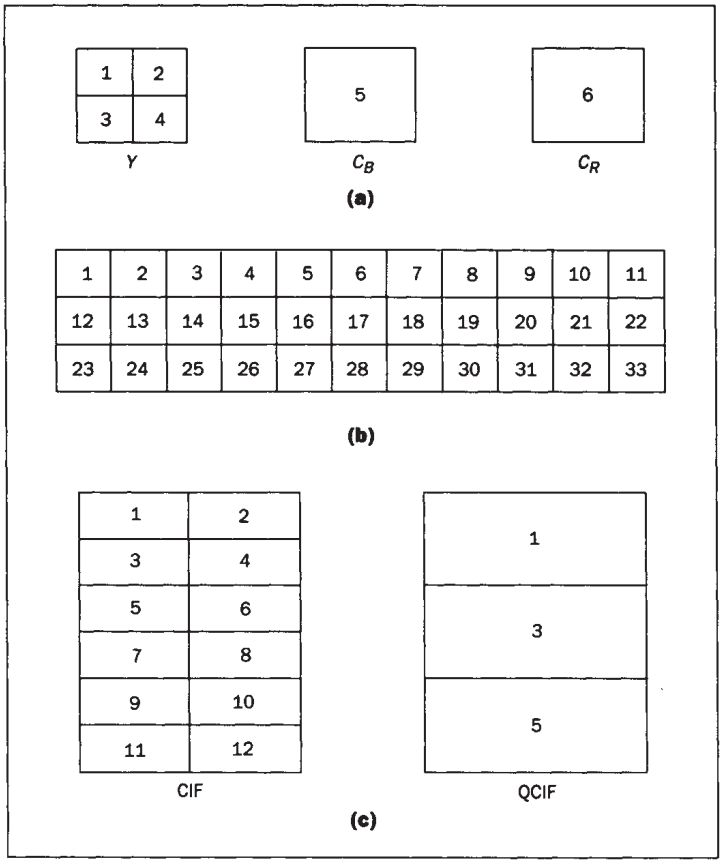


Fig. 8.5. Successive arrangement of (a) blocks in a Macroblock, (b) Macroblocks in a GOB, and (c) GOBs in a picture.

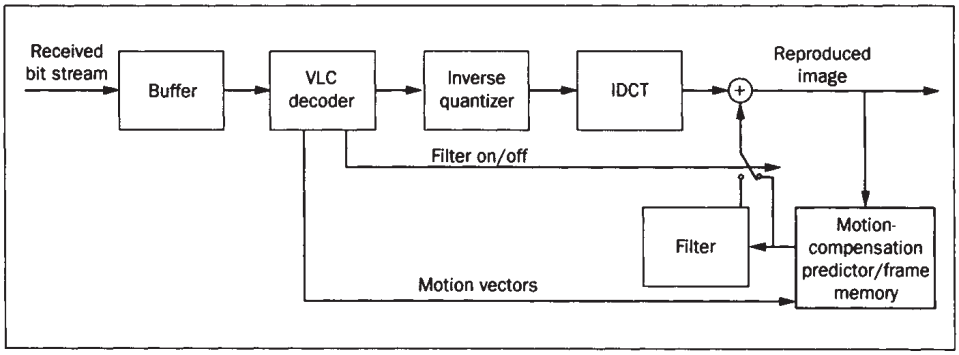


Fig. 8.6. A typical H.261 decoder.

- Picture start code (PSC) — a 20-bit pattern.
- Temporal reference (TR) — a 5-bit input frame number.
- Type information (PTYPE) — such as CIF/QCIF selection.
- Spare bits to be defined in later versions.

GOB Layer

At the GOB layer, a GOB header contains:

- Group of blocks start code (GBSC) — a 16-bit pattern.
- Group number (GN) — a 4-bit GOB address.
- Quantizer information (GQUANT) — Initial quantizer step size normalized to the range 1 to 31. At the start of a GOB, we set QUANT = GQUANT.
- Spare bits to be defined in later versions of the standard.

Next, come a number of MB layers. An 11-bit stuffing pattern can be inserted repetitively right after a GOB header or after a transmitted Macroblock.

Macroblock (MB) Layer

At the MB layer, the header contains:

- Macroblock address (MBA) — location of this MB relative to the previously coded MB inside the GOB. MBA equals one plus the number of skipped MBs preceding the current MB in the GOB.
- Type information (MTYPE) — 10 types in total.
- Quantizer (MQUANT) — normalized quantizer step size to be used until the next MQUANT or GQUANT. If MQUANT is received we set QUANT = MQUANT. Range is 1 to 31.
- Motion vector data (MVD) — differential displacement vector.
- Coded block pattern (CBP) — Indicates which blocks in the MB are coded. Blocks not coded are assumed to contain all zero coefficients.

Block Layer

The lowest layer is the block layer, consisting of quantized transform coefficients (TCOEFF), followed by the end of block (EOB) symbol. All coded blocks have the EOB symbol.

Not all header information need be present. For example, at the MB layer, if an MB is not Inter motion-compensated (as indicated by MTYPE), MVD does not exist. Also, MQUANT is optional. Most of the header information is coded using Variable Length Codewords.

Differential Motion Vectors	Code Words
-16 & 16	0000 0011 001
-15 & 17	0000 0011 011
-14 & 18	0000 0011 101
-13 & 19	0000 0011 111
-12 & 20	0000 0100 001
-11 & 21	0000 0100 011
-10 & 22	0000 0100 11
- 9 & 23	0000 0100 01
- 8 & 24	0000 0101 11
- 7 & 25	0000 0101
- 6 & 26	0000 1001
- 5 & 27	0000 1011
- 4 & 28	0000 111
- 3 & 29	0001
- 2 & 30	0011
- 1	011
0	1
1	10
2 & -30	0010
3 & -29	0001 0
4 & -28	0000 110
5 & -27	0000 1010
6 & -26	0000 1000
7 & -25	0000 0110
8 & -24	0000 0101 10
9 & -23	0000 0101 00
10 & -22	0000 0100 10
11 & -21	0000 0100 010
12 & -20	0000 0100 000
13 & -19	0000 0011 110
14 & -18	0000 0011 100
15 & -17	0000 0011 010

Fig. 8.7. Variable Length Codes for differential motion vectors. Since the motion vector range is limited to ±15 pels, only one of the differential vector values can be legal.

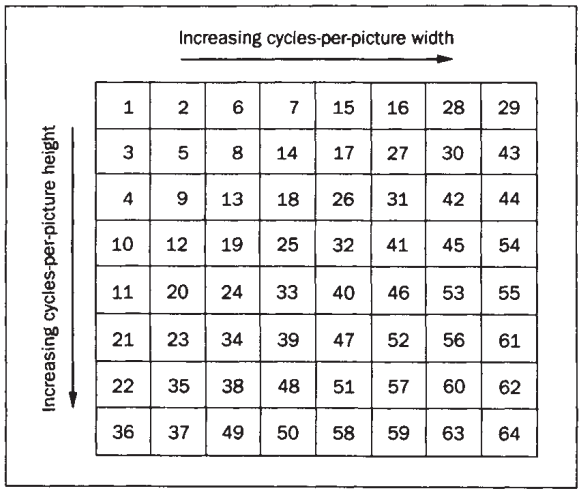


Fig. 8.8. Zigzag transmission order for transform coefficients.

Fig. 8.6 is a functional diagram of a typical H.261 decoder. The received video bit stream is first kept in the receiver buffer. The VLD decodes the compressed bits and distributes the decoded information to the elements that need that information. Further processing proceeds as described below.

There are essentially four types of coded MBs as indicated by MTYPE:

- Intra — original pels are transform-coded.
- Inter — frame difference pels (with zero-motion vectors) are coded. Skipped MBs are considered inter by default.
- Inter_MC — displaced (nonzero-motion vectors) frame differences are coded.
- Inter_MC_with_filter — the displaced blocks are filtered by a predefined loop filter, which may help reduce visible coding artifacts at very low bit rates.

Certain MB types in this list allow the optional transmission of MQUANT and TCOEFF information. The received MTYPE information controls various switches at the decoder to produce the correct combination.

A single-motion vector (horizontal and vertical displacement) is transmitted for one Inter_MC MB. That is, the four Y blocks, one C_B, and one C_R block all share the same motion vector. The range of motion vectors is ±15 Y pels with integer values. For color blocks, the motion vector is obtained by halving the transmitted vector and truncating the magnitude to an integer value.

Motion vectors are differentially coded using, in most cases, the motion vector of the MB to the left as a prediction. Zero is used as a prediction for the leftmost MBs of the GOB, and also if the MB to the left has no motion vector. Differential motion vectors are coded using the VLC of Fig. 8.7. Thus, using both MVD and MTYPE information, the predictor is able to choose the right pels for prediction.

The transform coefficients of either the original (Intra) or the differential (Inter) pels are ordered according to the zigzag scanning pattern in Fig. 8.8. These transform coefficients are selected and quantized at the encoder, and then coded using variable-length codewords (VLCs) and/or fixed-length codewords (FLC), depending on the values. Just as with JPEG, successive zeros between two nonzero coefficients are counted and called a *RUN*. The value of a transmitted nonzero quantized coefficient is called a *LEVEL*. The most likely occurring combinations of (RUN, LEVEL) are encoded with the VLC table shown in Fig. 8.9, with the sign bit terminating the RUN-LEVEL VLC codeword. The other combinations are coded with a 20-bit FLC consisting of a 6-bit ESCAPE code, 6 bits RUN, and 8 bits LEVEL. However, Intra DC coefficients use a different FLC. EOB is always appended to the last codeword, indicating the end of a block.

Run	Level	Code Word
0	1	1s (if first inter coefficient)
0	1	11s (otherwise)
0	2	0100 s
0	3	0010 1s
0	4	0000 110s
0	5	0010 0110s
0	6	0010 0001s
0	7	0000 0010 10s
0	8	0000 0001 1101s
0	9	0000 0001 1000s
0	10	0000 0001 0011s
0	11	0000 0001 0000s
0	12	0000 0000 1101s
0	13	0000 0000 1100s
0	14	0000 0000 1100s
0	15	0000 0000 1011s
1	1	011s
1	2	0001 10s
1	3	0010 0101s
1	4	0000 0011 00s
1	5	0000 0001 1011s
1	6	0000 0000 1011 0s
1	7	0000 0000 1010 1s
2	1	0101 s
2	2	0000 100s
2	3	0000 0010 11s
2	4	0000 0001 0100s
2	5	0000 0000 1010 0s
3	1	0011 1s
3	2	0010 0100s
3	3	0000 0001 1100s
3	4	0000 0000 1001 1s
4	1	0011 0s
4	2	0000 0011 11s
4	3	0000 0001 0010s
5	1	0001 11s
5	2	0000 0010 01s
5	3	0000 0000 1001 0s
6	1	0001 01s
6	2	0000 0001 1110s
7	1	0001 00s
7	2	0000 0001 0101s
8	1	0000 111s
8	2	0000 0001 0001s
9	1	0000 101s
9	2	0000 0000 1000 1s
10	1	0010 0111 s
10	2	0000 0000 1000 0s
11	1	0010 0011 s
12	1	0010 0010 s
13	1	0010 0000 s
14	1	0000 0011 10s
15	1	0000 0011 01s
16	1	0000 0010 00s
17	1	0000 0000 1111s
18	1	0000 0001 1010s
19	1	0000 0001 1001s
20	1	0000 0001 0111s
21	1	0000 0001 0110s
22	1	0000 0000 1111 1s
23	1	0000 0000 1111 0s
24	1	0000 0000 1110 1s
25	1	0000 0000 1110 0s
26	1	0000 0000 1101 1s

EOB 10

Escape 0000 01

Fig. 8.9. Variable Length Codes (VLCs) for zero-run lengths (RUN) and quantized coefficient values (LEVEL). The sign bit s terminates the VLC word. Other combinations are coded with a 20-bit fixed length code (FLC), consisting of the ESCAPE code, followed by a 6-bit RUN and an 8-bit LEVEL. The End of Block (EOB) symbol is inserted after the last codeword of every block.

The inverse quantizer or the reconstruction process for all the coefficients other than the intra DC is defined by the following formula:

$$REC = QUANT \times (2 \times LEVEL + Sign), \text{ for } odd\ QUANT \qquad (8.1)$$

$$REC = QUANT \times (2 \times LEVEL + Sign) - Sign, \text{ for } even\ QUANT$$

where *Sign* = +1, -1 or 0 depending on whether *LEVEL* is positive, negative or zero, respectively.

QUANT is the half step size determined by GQUANT or MQUANT, and REC is the reconstructed value of a quantized coefficient clipped to the range -2048 to 2047. Almost all the reconstruction levels are odd numbers to reduce problems of mismatch between encoders and decoders from different manufacturers. The intra-DC coefficient is uniformly quantized with a fixed step size of 8, and coded with 8 bits.

The standard requires a compatible IDCT (inverse DCT) to be close to the ideal 64-bit floating point IDCT. H.261 specifies a measuring process for checking a valid IDCT. The error in pel values between the ideal IDCT and the IDCT under test must be less than certain allowable limits given in the standard, e.g., peak error ≤ 1, mean error ≤ 0.0015, and mean square error ≤ 0.02.

A few other items are also required by the standard. One of them is the image-block updating rate. To prevent mismatched IDCT error as well as channel error propagation, every MB should be intra-coded at least once in every 132 transmitted picture frames.

The contents of the transmitted video bit stream must also meet the requirements of the *hypothetical reference decoder* (HRD). For CIF pictures, every coded frame is limited to fewer than 256 Kbits; for QCIF, the limit is 64 Kbits, where K = 1024. The HRD receiving buffer size is $B + 256$ Kbits, where $B = 4 \times R_{max} / 29.97$ and R_{max} is the maximum connection (channel) rate. At every picture interval (1/29.97 sec), the HRD buffer is examined. If at least one complete coded picture is in the buffer, then the earliest picture bits are removed from the buffer and decoded. The buffer occupancy, right after the above bits have been removed, must be less than B.

Note that this definition of HRD allows for varying the delay during the video coding. If the encoder assigns a relatively large number of bits to a particular frame, the decoder must repeat a few frames waiting for the arrival of those bits, thus increasing the delay. Conversely, if the encoder assigns relatively fewer bits to a frame, the decoder can decode early, thus reducing the delay.

8.4 Multipoint Features of P*64

Several signalling features have been provided in P*64 to facilitate video conferencing between several separated points. For example, the *Freeze Picture* signal requests the decoder to stop decoding and to continually repeat the most recently received frame in preparation for a disruption in the video bit stream. A

Freeze Release signal tells the decoder that decoding can resume with the reception of the current picture. A *Fast Update* signal tells the encoder to encode the entire next frame using only Intra MBs.

These signals are used typically by a centrally located Multipoint Control Unit (MCU) that interconnects the participants in the videoconference. For example, if it is desired that all participants see the person who is talking, then the MCU must detect whenever there is a new speaker. It then issues a Freeze Picture request to all decoders, routes the speaker's encoder output bit stream to all decoders, issues a Fast Update signal to the speaker's encoder which issues a Freeze Release signal to all decoders.

8.5 Encoder Constraints and Options

Fig. 8.10 shows a typical encoder structure. The input standard color components, e.g., YC_BC_R of Sec. 2.2.5, are spatially and temporally filtered so that they can be resampled in an internationally compatible common intermediate format. The spatio-temporal filter may also adaptively effect additional filtering, as indicated by the data-rate control, in order to allow for reduced resolution or a reduced frame rate for actual coding. Such control information must also be passed to the multiplexer and sent to the decoder to enable proper interpretation of the received bits.

The uncoded pels from the current frame and the coded pels from the frame memory both pass to the motion estimator, which then calculates one displacement vector per luminance motion estimation block, i.e., 16×16 Y pels. The motion estimator would probably use one of the block matching methods of Section 5.2.3 in its calculation and produce a motion vector with single pel displacement accuracy and a maximum displacement range of ±15 vertically and horizontally. Chrominance blocks use half the displacement of the corresponding luminance blocks, thus eliminating the need to compute and send chrominance displacements.

The motion estimation then passes to the predictor, which applies the indicated displacement to the contents of the frame memory and passes the result to the subtractor. The output of the subtractor is the motion compensated frame difference, also known as the Displaced Frame Difference (DFD), which is then partitioned into two-dimensional 8×8 blocks for transformation.

The data-rate control unit must compare the uncoded pels with the DFD pels in order to decide whether the MB is to be coded Intra or Inter. Whichever minimizes a simple sum of magnitudes is often a sufficiently good decision.

If the decision is Inter and the motion vector in nonzero, the data-rate control unit must decide whether to invoke the loop filter. A simple threshold on the sum of DFD magnitudes is often sufficient.

The transform coefficients are then adaptively quantized, variable word-length coded and passed to the multiplexer for transmission. All coefficients (except the Intra DC coefficient) are quantized with the same quantizer shown in

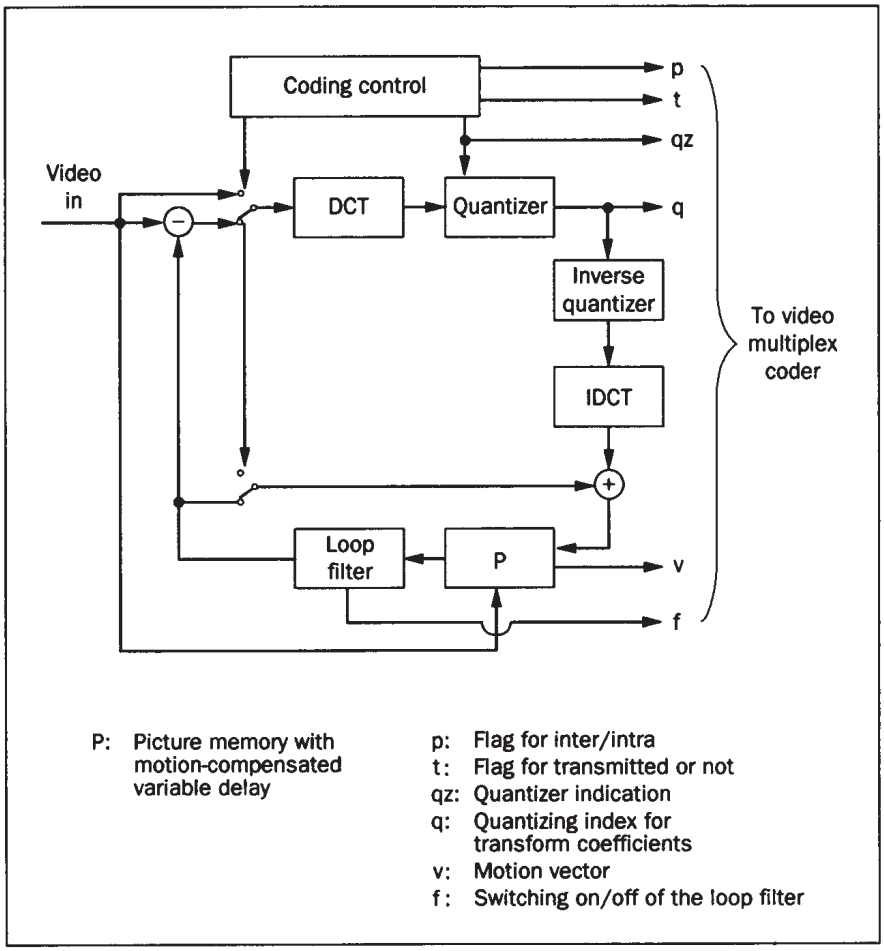


Fig. 8.10. Generic form of sub-primary rate coder. Input is assumed to be color components of the local standard. The spatio-temporal filter enables resampling to a common intermediate format. A subsampler (not shown) adaptively adjusts the frame rate, as instructed by the data-rate control. Motion compensated frame differences are transformed in 8×8 blocks, adaptively quantized under control of the buffer fullness, variable word-length coded, multiplexed with the side information and fed to a buffer (not shown) to await transmission. The coded image is reconstructed, fed to a frame memory and thence to a predictor and motion estimator to be used in coding the next frame. SI is side information that is sent to enable correct decoding at the receiver.

Fig. 8.11, whose step size $L = 2 \times QUANT$ is variable from MB to MB, as set by the data-rate control. The quantized coefficients are then inverse transformed and added to the prediction (zero for Intra) to form a reconstructed picture that is fed to the frame memory to be used in coding the next frame.

The output of the multiplexer is fed to a buffer that smooths the data rate prior to sending to the channel coder. The channel coder computes parity bits for a (511,493) BCH code, adds a framing bit and sends the result to the audio/video/signalling multiplexer and hence to the ISDN channel.

For the purpose of this discussion, the elements inside a standard compatible encoder can be classified, based on their functionalities, into two categories:

- The *basic coding operation* units, such as motion estimator, quantizer, transform, and variable-word-length encoder (VLE).
- The *coding parameter decision* units, such as the coding control in Fig. 8.10. These units select the parameter values of the basic operation units, including motion vectors, quantization step size, and picture frame rate.

Although H.261 does not explicitly specify a standard encoder, most basic operation elements are strongly constrained by the standard. However, other crucial elements, such as the parameter decision unit, are still open to the design engineers. We briefly outline some observations below.

The VLE is pretty much fixed because the VLC tables are given. The forward DCT is not specified, but it is expected that the IDCT inside the encoder matches fairly closely the decoder IDCT. Moreover, the encoder forward DCT should match its own IDCT.

The control examines the quantized coefficients to see if they are all or nearly all zero. If so, that block may not be coded. If all six blocks of an Inter (no MC) MB have insignificant coefficients, the entire MB may be skipped.

Also, the control can force the predictor to output zeros, thus causing the transform to take place on the pels themselves (Intra) rather than on the displaced frame differences (Inter). This intraframe coding mode might be used following a scene change or for forced updating to accommodate for transmission bit errors.

The control decides which spatial and temporal resolution reductions are appropriate, and signals the spatio-temporal filter accordingly. With lower spatial resolution, fewer transform coefficients need be sent per block. In fact, some implementations effect the spatial filtering by simply reducing the number of transmitted coefficients. However, with 8×8 blocks, this can cause deleterious effects due to the visibility of block boundaries as the bit-rate declines.

Because the inverse quantizer (IQ) is defined at the decoder, variations of the encoder quantizer are quite limited. From a theoretical viewpoint, however, it is not necessary for the decision levels of the encoder quantizer to be in the

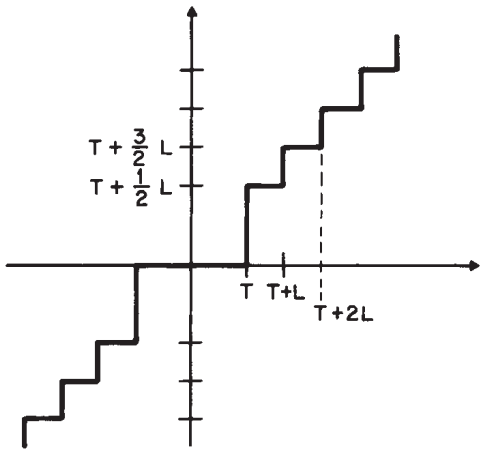


Fig. 8.11. Quantizer used for all transform coefficients except Intra DC. The characteristic is uniform except for the dead zone around zero. (See Eq. 8.1) This dead zone helps to decrease sensitivity to noise in the input. The step size $L = 2 \times QUANT$ varies from 2 to 62 in intervals of 2. To alleviate mismatch, representation levels are made odd by decrementing toward zero, if necessary.

middle of two reconstruction levels. Also, encoder designers determine the criterion (a fixed or an adaptive threshold, for example) for selecting transform coefficients to be coded.

If motion compensation is selected, the motion estimator must be able to produce one motion vector for the entire MB. However, the displacement-estimating algorithm is completely open to the designer. Block matching is a popular scheme, but many other block-motion estimation algorithms exist, such as gradient-based methods. Even within the category of block-matching algorithms, there are several variations, such as hierarchical-motion estimation.

Because of the HRD model required by the standard, the encoder output bits of every coded frame must be regulated carefully. For example, many successive low bit frames may violate the HRD requirement.

For video teleconferencing the size of the buffer must be limited to avoid excessive transmission delay. For a channel rate R and Buffer Size BS the one-way delay due to buffering is BS/R. Thus, for a one-way delay of 250 msec the buffer size should not exceed 28 kbits for Basic Rate ISDN (112 kbits/s video) and approximately 80 kbits for the H0 channel (320 kbits/s video). Although the decoder buffer size specified by H.261 is much larger than these numbers, it is the encoder that decides how much of the buffer to use and, therefore, how much delay is incurred by coding.

Although individual basic coding elements may affect the overall coding performance, the most critical and global influence on the encoder performance comes from the parameter decision units. The questions they have to answer are:

- How many frames should be transmitted, or conversely, how many should be skipped?
- What MTYPE should each Macroblock use?
- What is the proper quantization step size?
- How do we control the buffer fullness so that it does not produce long delay and does not violate the HRD requirements?

Researchers have studied many of the above questions, and various implementations are in the market. It is important to keep the hardware relatively simple for practical applications. Thus, different products will make different trade-offs of performance and cost. Since the H.261 standard does not specify the encoder, a wide latitude in capabilities is possible. Moreover, as time goes on and circuit costs decline, the performance of codecs will improve even though the standard changes little.

References

8.1 ITU-T (CCITT), *Recommendation H.261---Video Codec for Audiovisual Services at p*64 kbit/s*, Geneva, August 1990.

Questions for Understanding

8.1 What is the luminance pel sampling rate for CIF? QCIF?

8.2 What is the purpose of stuffing information?

8.3 If the first two MBs in a GOB have MVs (2,-2) and (-3,4), what code words are used to code the MVs?

8.4 What are the MBAs for the above example?

8.5 If the first five quantized DCT coefficients of an Inter block have levels 20,0,0,-3,22 what code words are used to code them?

8.6 If QUANT = 4, what are the reconstructed coefficient values for the above example?

8.7 If QUANT = 5, what are the reconstructed coefficient values for the above example?

8.8 If QUANT=5 and the first five unquantized DCT coefficients of an Inter block have values -27, 33, 138, -100 and 86, what are the quantized levels? What are the reconstructed values? What are the code words?

8.9 Suppose an encoder codes the first four frames using 4000, 3000, 2000 and 1000 bits, respectively. At a bit rate of 64 kbits/sec, plot the fullness of the HRD buffer.

8.10 In the above example, plot the fullness of the encoder buffer assuming the encoder produces a constant number of bits per pel.

8.11 In the above example, plot the fullness of the encoder buffer assuming the encoder produces all the pel bits at the top of the picture.

8.12 In the above example, plot the fullness of the encoder buffer assuming the encoder produces all the pel bits at the bottom of the picture.

8.13 How does the HRD buffer fullness change in the above three examples?

9

Coding for Entertainment Video - ISO MPEG

9.1 Moving Picture Experts Group (MPEG)

This group was chartered by the International Standards Organization (ISO) to standardize a coded representation of video and associated audio suitable for digital storage and transmission media. Digital storage media include magnetic computer disks, optical compact disk read-only-memory (CD-ROM), digital audio tape (DAT), etc. Transmission media include telecommunications networks, home coaxial cable TV (CATV), and over-the-air digital video. The group's goal has been to develop a *generic* coding standard that can be used in many digital video implementations. Thus, particular applications will typically require some further specification and refinement to be useful. As of this writing, MPEG has produced two standards, known colloquially as MPEG1 and MPEG2. Work also continues on methods and applications for future coding standards.

9.2 Video Coding at 1.5 Mbits/s (MPEG1)

MPEG1[9.1–9.3] is an international standard for the coded representation of digital video and associated audio at bit-rates up to about 1.5 Mbits/s. The colloquial name of the standard comes from the experts group that developed the standard, MPEG. The official name is ISO/IEC 11172.

If video is coded at about 1.1 Mbits/s and stereo audio is coded at 128 kbits/s per channel, then the total audio/video digital signal will fit onto the CD-ROM bit-rate of approximately 1.4 Mbits/s as well as the North American ISDN

This chapter contains excerpts taken with permission from AT&T Technical Journal[7.17].

Primary Rate (23 B-channels) of 1.47 Mbits/s. The specified bit-rate of 1.5 Mbits/s is not a hard upper limit. In fact, MPEG1 allows rates as high as 100 Mbits/s. However, during the course of MPEG1 algorithm development, coded image quality was optimized at a rate of 1.1 Mbits/s using progressive* scanned color images. Two *Source Input Formats* (SIF) were used for optimization. One corresponding to NTSC was 352 pels, 240 lines, 29.97 frames/sec. The other corresponding to PAL, was 352 pels, 288 lines, 25 frames/sec. SIF uses 2:1 color subsampling, both horizontally and vertically, in the same 4:2:0 format as H.261.

The MPEG1 standard currently has four parts (others will follow):

- Part 1 "Systems" concerns the synchronization and multiplexing of video, audio and ancillary data.
- Part 2 "Video" describes the coded video bitstream syntax and semantics.
- Part 3 "Audio" describes the coded audio bitstream syntax and semantics.
- Part 4 "Compliance Testing" describes tests for verifying if bitstreams and/or decoders meet the requirements specified in the first three parts.

An overview of the video part of the MPEG1 standard follows.

9.2.1 Requirements of the MPEG1 Video Standard

Uncompressed digital video requires an extremely high transmission bandwidth. Digitized NTSC resolution video, for example, has a bit-rate of approximately 100 Mbits/s. With digital video, compression is necessary to reduce the bit-rate to suit most applications. The required degree of compression is achieved by exploiting the spatial and temporal redundancy present in a video signal. However, the compression process is inherently lossy, and the signal reconstructed from the compressed bit stream is not identical to the input video signal. Compression typically introduces some artifacts into the decoded signal.

The primary requirement of the MPEG1 video standard is that it should achieve the highest possible quality of the decoded motion video at a given bit-rate. In addition to picture quality under normal play conditions, different applications have additional requirements. For instance, multimedia applications may require the ability to randomly access and decode any single video picture† in the bitstream. Also, the ability to perform fast search directly on the bit stream, both forward and backward, is extremely desirable if the storage medium has ''seek'' capabilities. It is also useful to be able to edit

---

* The term *NonInterlaced* is also used for progressively scanned pictures.

† Frames and pictures are synonymous in MPEG1.

compressed bit streams directly while maintaining decodability. And finally, a variety of video formats should be supported.

9.2.2 Compression Algorithm Overview

Both spatial and temporal redundancy reduction are needed for the high compression requirements of MPEG1. Most techniques used by MPEG1 have been described in previous chapters.

9.2.2a Exploiting spatial redundancy

The compression approach of MPEG1 video uses a combination of the ISO JPEG (still image) and CCITT H.261 (videoconferencing) standards. Because video is a sequence of still images, it is possible to achieve some compression using techniques similar to JPEG. Such methods of compression are called intraframe coding techniques, where each picture of video is individually and independently compressed or encoded. Intraframe coding exploits the spatial redundancy that exists between adjacent pels of a picture. Pictures coded using only intraframe coding are called *I-pictures*.

As in JPEG and H.261, the MPEG1 video-coding algorithm employs a Block-based two-dimensional DCT. A picture is first divided into 8×8 Blocks of pels, and the two-dimensional DCT is then applied independently on each Block. This operation results in an 8×8 Block of DCT coefficients in which most of the energy in the original (pel) Block is typically concentrated in a few low-frequency coefficients. The coefficients are scanned and transmitted in the same zigzag order as JPEG and H.261.

A quantizer is applied to the DCT coefficients, which sets many of them to zero. This quantization is responsible for the lossy nature of the compression algorithms in JPEG, H.261 and MPEG1 video. Compression is achieved by transmitting only the coefficients that survive the quantization operation and by entropy-coding their locations and amplitudes.

9.2.2b Exploiting temporal redundancy

Many of the interactive requirements can be satisfied by intraframe coding. However, as in H.261, the quality achieved by intraframe coding alone is not sufficient for typical video signals at bit-rates around 1.1 Mbits/s.

Temporal redundancy results from a high degree of correlation between adjacent pictures. The MPEG1 algorithm exploits this redundancy by computing an interframe difference signal called the *prediction error*. In computing the prediction error, the technique of motion compensation is employed to correct for motion. A Macroblock (MB) approach is adopted for motion compensation.

In unidirectional motion estimation, called *Forward Prediction*, a *Target* Macroblock in the picture to be encoded is matched with a set of blocks of the same size in a past picture called the *Reference* picture. As in H.261 the

Macroblock in the Reference picture that "best matches" the Target Macroblock is used as the *Prediction* Macroblock. The prediction error is then computed as the difference between the Target Macroblock and the Prediction Macroblock.* The position of this best-matching Prediction Macroblock is indicated by a motion vector that describes the displacement between it and the Target Macroblock. The motion vector information is also encoded and transmitted along with the prediction error. Pictures coded using Forward Prediction are called *P-pictures.*

The prediction error itself is transmitted using the DCT-based intraframe encoding technique summarized above. In MPEG1 video (as in H.261), the Macroblock size for motion compensation is chosen to be 16×16, representing a reasonable trade-off between the compression provided by motion compensation and the cost associated with transmitting the motion vectors.

9.2.2c Bidirectional temporal prediction

Bidirectional temporal prediction, also called Motion-Compensated Interpolation, is a key feature of MPEG1 video. Pictures coded with Bidirectional prediction use two Reference pictures, one in the past and one in the future. A Target Macroblock in bidirectionally coded pictures can be predicted by a Macroblock from the past Reference picture (*Forward Prediction*), or one from the future Reference picture (*Backward Prediction*), or by an average of two Macroblocks, one from each Reference picture (*Interpolation*). In every case, a Prediction Macroblock from a Reference picture is associated with a motion vector, so that up to two motion vectors per MB may be used with Bidirectional prediction. Motion-Compensated Interpolation for a Macroblock in a Bidirectionally predicted picture is illustrated in Fig. 9.1.

Pictures coded using Bidirectional Prediction are called *B-pictures.* Pictures that are Bidirectionally predicted are never themselves used as Reference pictures, i.e., Reference pictures for B-pictures must be either P-pictures or I-pictures. Similarly, Reference pictures for P-pictures must also be either P-pictures or I-pictures.

Bidirectional prediction provides a number of advantages. The primary one is that the compression obtained is typically higher than can be obtained from Forward (unidirectional) prediction alone. To obtain the same picture quality, Bidirectionally predicted pictures can be encoded with fewer bits than pictures using only Forward prediction.

---

* Prediction MBs do not, in general, align with coded MB boundaries in the Reference frame.

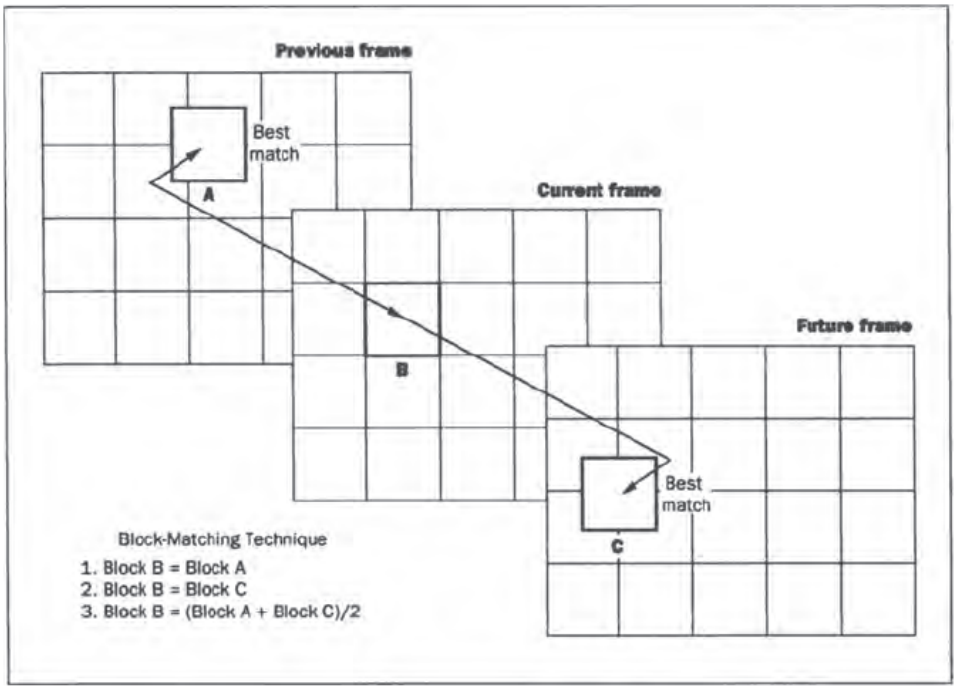


Fig. 9.1 Motion-Compensated Interpolation using Bi-directional prediction.

Table 9.1 Six Headers of MPEG1 Video Bit-Stream Syntax

Syntax Header	Functionality
Sequence	Context unit
Group of pictures	Random access unit
Picture	Primary coding unit
Slice	Resynchronization unit
Macroblock	Motion compensation unit
Block	DCT unit

However, Bidirectional prediction does introduce extra delay in the encoding process, because pictures must be encoded out of sequence. Further, it entails extra encoding complexity because Macroblock matching (the most computationally intensive encoding procedure) has to be performed twice for each Target Macroblock, once with the past Reference picture and once with the future Reference picture.

9.2.3 Features of the MPEG1 Bit-Stream Syntax

The MPEG1 video standard specifies the *syntax* and *semantics* of the compressed bit stream produced by the video encoder. The standard also specifies how this bit stream is to be parsed and decoded to produce a decompressed video signal.

Many encoder procedures are not specified, i.e., different algorithms can be employed at the encoder as long as the resulting bit stream is consistent with the specified syntax. For example, the details of the motion estimation matching procedure are not part of the standard. However, as with H.261 there is a strong limitation on the variation in bits/picture in the case of constant bit-rate operation. This is enforced through a *Video Buffer Verifier* (VBV), which corresponds to the Hypothetical Reference Decoder of H.261. Any MPEG1 bit stream is prohibited from overflowing or underflowing the buffer of this VBV. Thus, unlike H.261, there is no picture skipping allowed in MPEG1.

The bit-stream syntax is flexible in order to support the variety of applications envisaged for the MPEG1 video standard. To this end, the overall syntax is constructed in a hierarchy* of several Headers, each performing a different logical function. The different Headers in the syntax and their use are illustrated in Table 9.1.[9.3]

Video Sequence Header

The outermost Header is called the *Video Sequence* Header, which contains basic parameters such as the size of the video pictures, Pel Aspect Ratio (PAR), picture rate, bit-rate, assumed VBV buffer size and certain other global parameters. This Header also allows for the optional transmission of JPEG style Quantizer Matrices, one for Intra coded pictures and one for Non-Intra coded pictures. Unlike JPEG, if one or both quantizer matrices are not sent, default values are defined. These are shown in Fig. 9.2. Private *user data* can also be sent in the Sequence Header as long as it does not contain a *Start Code Header* , which MPEG1 defines as a string of 23 or more zeros.

---

* As in H.261, MPEG1 uses the term *Layers* for this hierarchy. However, Layer has another meaning in MPEG2. Thus, to avoid confusion we will not use Layers in this section.

8	16	19	22	26	27	29	34
16	16	22	24	47	49	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Intra

16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

Inter

Fig. 9.2 Default Quantizer Matrices for Intra and NonIntra Pictures.

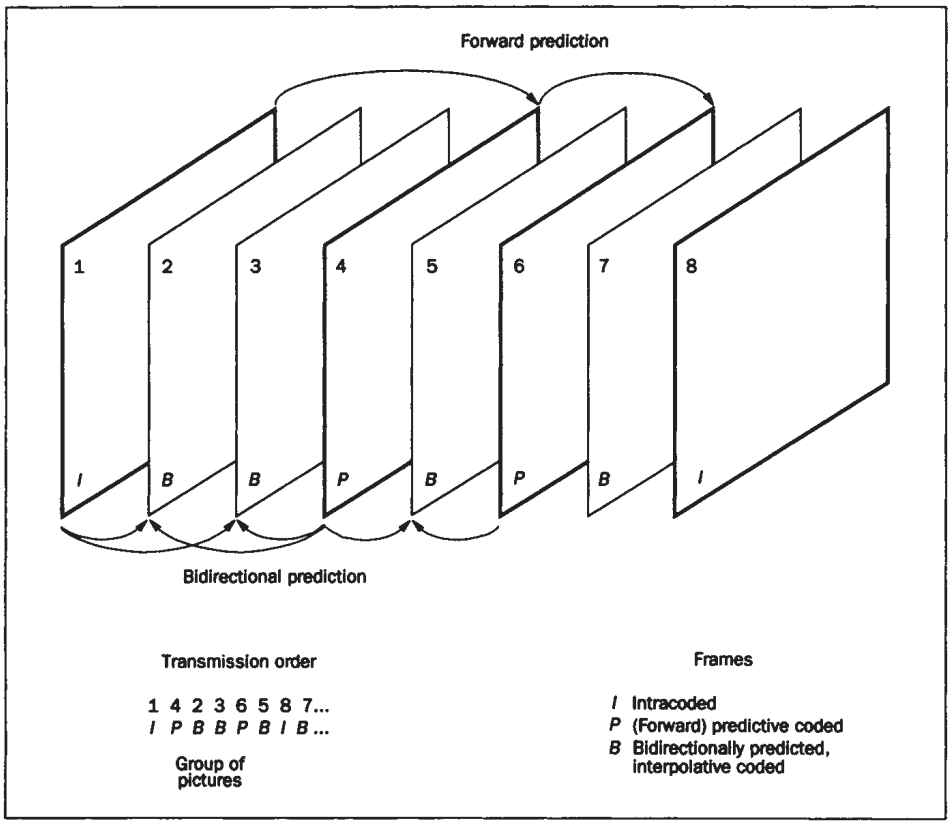


Fig. 9.3 Group of pictures. Video pictures are not transmitted in display order.

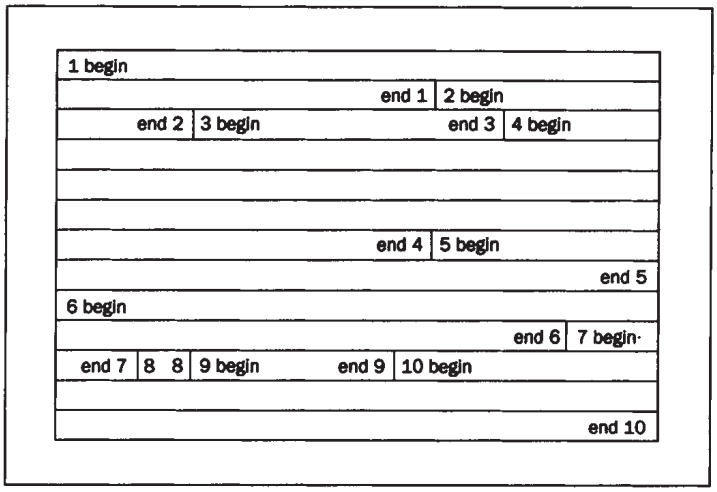


Fig. 9.4 Possible arrangement of Slices in a 256 × 192 picture.

Group of Pictures (GOP) Header

Below the Video Sequence Header is the *Group of Pictures* (GOP) Header, which provides support for random access, fast search, and editing. A sequence of transmitted video pictures is divided into a series of GOPs, where each GOP contains an intra-coded picture (I-picture) followed by an arrangement of Forward predictive-coded pictures (P-pictures) and Bidirectionally predicted pictures (B-pictures). Fig. 9.3 shows a GOP example with six pictures, 1 to 6. This GOP contains I-picture 1, P-pictures 4 and 6, and B-pictures 2, 3 and 5. The encoding/transmission order of the pictures in this GOP is shown at the bottom of Fig. 9.3. B-pictures 2 and 3 are encoded after P-picture 4, using P-picture 4 and I-picture 1 as reference. Note that B-picture 7 in Fig. 9.3 is part of the next GOP because it is encoded after I-picture 8.

Random access and fast search are enabled by the availability of the I-pictures, which can be decoded independently and serve as starting points for further decoding. The MPEG1 video standard allows GOPs to be of arbitrary structure and length. The GOP Header is the basic unit for editing an MPEG1 video bit stream.

Picture Header

Below the GOP is the *Picture* Header, which contains the type of picture that is present, e.g., I, P or B, as well as a *Temporal Reference* indicating the position of the picture in display order within the GOP. It also contains a parameter called *vbv_delay* that indicates how long to wait after a random access before starting to decode. Without this information, a decoder buffer could underflow or overflow following a random access.

Slice Header

A *Slice* is a string of consecutive Macroblocks of arbitrary length running from left to right and top to bottom across the picture. The Slice Header is intended to be used for re-synchronization in the event of transmission bit errors. Prediction registers used in the differential encoding of motion vectors and DC Intra coefficients are reset at the start of a Slice. It is again the responsibility of the encoder to choose the length of each Slice depending on the expected bit error conditions. The first and last MBs of a Slice cannot be skipped MBs, and gaps are not allowed between Slices. The Slice Header contains the vertical position of the Slice within the picture, as well as a *quantizer_scale* parameter (corresponding to GQUANT in H.261). Fig. 9.4 shows an example in which Slice lengths vary throughout the picture.

Macroblock Header

The Macroblock (MB) is the 16×16 motion compensation unit. In the Macroblock Header, the horizontal position (in MBs) of the first MB of each Slice is coded with the MB Address VLC. The positions of additional

transmitted MBs are coded differentially with respect to the most recently transmitted MB, also using the MB Address VLC. Skipped MBs are not allowed in I-pictures. In P-pictures, skipped MBs are assumed NonIntra with zero coefficients and zero motion vectors. In B-pictures, skipped MBs are assumed NonIntra with zero coefficients and motion vectors the same as the previous MB. Also included in the Macroblock Header are MB Type (Intra, NonIntra, etc.), *quantizer_scale* (corresponding to MQUANT in H.261), motion vectors and coded block pattern. As with other Headers, these parameters may or may not be present, depending on MB Type.

Block

A *Block* consists of the data for the quantized DCT coefficients of an 8×8 Block in the Macroblock. It is VLC coded as described in the next sections. For noncoded Blocks, the DCT coefficients are assumed to be zero.

9.2.4 Quantization of DCT Coefficients

As with JPEG, MPEG1 Intra coded blocks have their DC coefficients coded differentially with respect to the previous block of the same YCbCr type, unless the previous block is nonIntra or belongs to another Slice, in which case the DC coefficient is coded differentially with respect to the value 8*128=1024. The range of unquantized Intra DC coefficients is 0 to 8*255, which means the range of differential values is -2040 to 2040. Intra DC Differentials are quantized with a uniform mid-step quantizer with fixed step size 8. Thus, quantization is normally implemented simply by dividing by 8 and rounding to the nearest integer. Quantized Levels have a range of -255 to 255.

Intra AC coefficients are quantized with a nearly uniform mid-step quantizer having variable step size. The slight nonuniformity is because representation Levels are constrained to odd integer values by decrementing toward zero by one, if necessary. This constraint alleviates mismatch. For coefficient $c[m][n]$ the step size is given by

$$(2*quantizer\_scale*IntraQmatrix[m][n])/16, \tag{9.1}$$

where *quantizer_scale* is set by a Slice or MB Header and $IntraQmatrix[m][n]$ is the Intra quantizer matrix value for DCT frequency $[m][n]$. Again, quantization is usually performed by division and rounding.

All NonIntra coefficients are quantized in the same way as in H.261 (See Eq. 8.1) using the stepsize

$$2*QUANT = (2*quantizer\_scale*NonIntraQmatrix[m][n])/16, \tag{9.2}$$

where $NonIntraQmatrix[m][n]$ is the NonIntra quantizer matrix value for DCT frequency $[m][n]$. In this case, the quantizer is nonuniform, with a *dead zone* of

Differential DC	SIZE	VLC Luminance	VLC Chrominance	VLI
-255 to -128	8	1111 110	1111 1110	00000000 to 01111111
-127 to -64	7	1111 10	1111 110	0000000 to 0111111
-63 to -32	6	1111 0	1111 10	000000 to 011111
-31 to -16	5	1110	1111 0	00000 to 01111
-15 to -8	4	110	1110	0000 to 0111
-7 to -4	3	101	110	000 to 011
-3 to -2	2	01	10	00 to 01
-1	1	00	01	0
0	0	100	00	
1	1	00	01	1
2 to 3	2	01	10	10 to 11
4 to 7	3	101	110	100 to 111
8 to 15	4	110	1110	1000 to 1111
16 to 31	5	1110	1111 0	10000 to 11111
32 to 63	6	1111 0	1111 10	100000 to 111111
64 to 127	7	1111 10	1111 110	1000000 to 1111111
128 to 255	8	1111 110	1111 1110	10000000 to 11111111

Fig. 9.5 Variable Length Codes (VLC) and Variable Length Integers (VLI) for quantized Intra DC differential values, which have a range of -255 to 255.

Table 9.2 Bounds for MPEG1 Constrained Parameter Bit-Streams (K=1024)

Parameter	Upper Bound
horizontal size	768 pels
vertical size	576 lines
number of MBs	396
MB rate	396*25 MB/s
frame rate	30 Hz
VBV Buffer	40 Kbytes
bit-rate	1.856 Mbits/s

half a step size around zero, as shown in Fig. 8.11. Thus, coefficient values should be shifted toward zero by at least *QUANT* prior to dividing by the step size 2**QUANT* and rounding.

Quantized Levels have a range of -255 to 255. In principle, this could be enforced by clipping after division by the step size. However, clipping may cause unacceptable picture distortion. A better solution is usually to increase the step size.

9.2.5 Variable Length Coding of Quantized DCT Coefficients

The Intra DC is coded similar to JPEG. Differential DC levels are coded as a VLC size followed by a VLI DIFF, where the VLC indicates the size in bits of the VLI. The VLCs and VLIs are given in Fig. 9.5 for quantized Intra DC DIFFs, which have a range of -255 to 255.

The remaining coefficients, including nonIntra DC, are coded in zigzag order using a combined runlength+level VLC, as in H.261. In fact, the first portion of the MPEG1 runlength-level VLC is exactly the same as in H.261. The remainder is shown in Fig. 9.6. Runlength-Level combinations that are not in the VLC are coded using a 6-bit Escape, 6-bit FLC for Runlength, an 8-bit FLC for Levels in the range -127 to 127 and a 16-bit FLC for remaining Levels. The 6-bit and 8-bit FLCs, as well as the Escape are the same as H.261. Quantized Levels have a range of -255 to 255. Every transmitted block ends with EOB.

9.2.6 Reconstruction of Quantized DCT Coefficients

Intra DC Differential values are obtained by simple multiplication of received Levels by 8. Intra AC coefficient reconstruction is complicated by the mismatch requirement of only odd integer values. The complete reconstruction of Intra AC coefficients is

$$\begin{aligned}&\tilde{c}[m][n]=(2*Level[m][n]*quantizer\_scale*IntraQmatrix[m][n])/16\\&\text{if}(\tilde{c}[m][n]\,\&\,1 == 0)\,\tilde{c}[m][n]-=Sign[m][n]\end{aligned} \tag{9.3}$$

where *Sign* = +1, -1 or 0 depending on whether *Level* is positive, negative or zero, respectively. All NonIntra coefficients are reconstructed by

$$\begin{aligned}&\tilde{c}[m][n]=((2*Level[m][n]\\&\quad+Sign[m][n])*quantizer\_scale*NonIntraQmatrix[m][n])/16\\&\text{if}(\tilde{c}[m][n]\,\&\,1 == 0)\;\tilde{c}[m][n]-=Sign[m][n]\end{aligned} \tag{9.4}$$

Values are then clipped to the range -2048 to 2047.

Run	Level	Code Word	Run	Level	Code Word
0	16	8*0 0111 11s	1	8	8*0 0011 111s
0	17	8*0 0111 10s	1	9	8*0 0011 110s
0	18	8*0 0111 01s	1	10	8*0 0011 101s
0	19	8*0 0111 00s	1	11	8*0 0011 100s
0	20	8*0 0110 11s	1	12	8*0 0011 011s
0	21	8*0 0110 10s	1	13	8*0 0011 010s
0	22	8*0 0110 01s	1	14	8*0 0011 001s
0	23	8*0 0110 00s	1	15	8*0 0001 0011s
0	24	8*0 0101 11s	1	16	8*0 0001 0010s
0	25	8*0 0101 10s	1	17	8*0 0001 0001s
0	26	8*0 0101 01s	1	18	8*0 0001 0000s
0	27	8*0 0101 00s	6	3	8*0 0001 0100s
0	28	8*0 0100 11s	11	2	8*0 0001 1010s
0	29	8*0 0100 10s	12	2	8*0 0001 1001s
0	30	8*0 0100 01s	13	2	8*0 0001 1000s
0	31	8*0 0100 00s	14	2	8*0 0001 0111s
0	32	8*0 0011 000s	15	2	8*0 0001 0110s
0	33	8*0 0010 111s	16	2	8*0 0001 0101s
0	34	8*0 0010 110s	27	1	8*0 0001 1111s
0	35	8*0 0010 101s	28	1	8*0 0001 1110s
0	36	8*0 0010 100s	29	1	8*0 0001 1101s
0	37	8*0 0010 011s	30	1	8*0 0001 1100s
0	38	8*0 0010 010s	31	1	8*0 0001 1011s
0	39	8*0 0010 001s			
0	40	8*0 0010 000s			

8*0 = 0000 0000

Fig. 9.6 Second half of MPEG1 VLC for RUNS and LEVELS. The first half is the same as for H.261, as shown in Fig. 8.9.

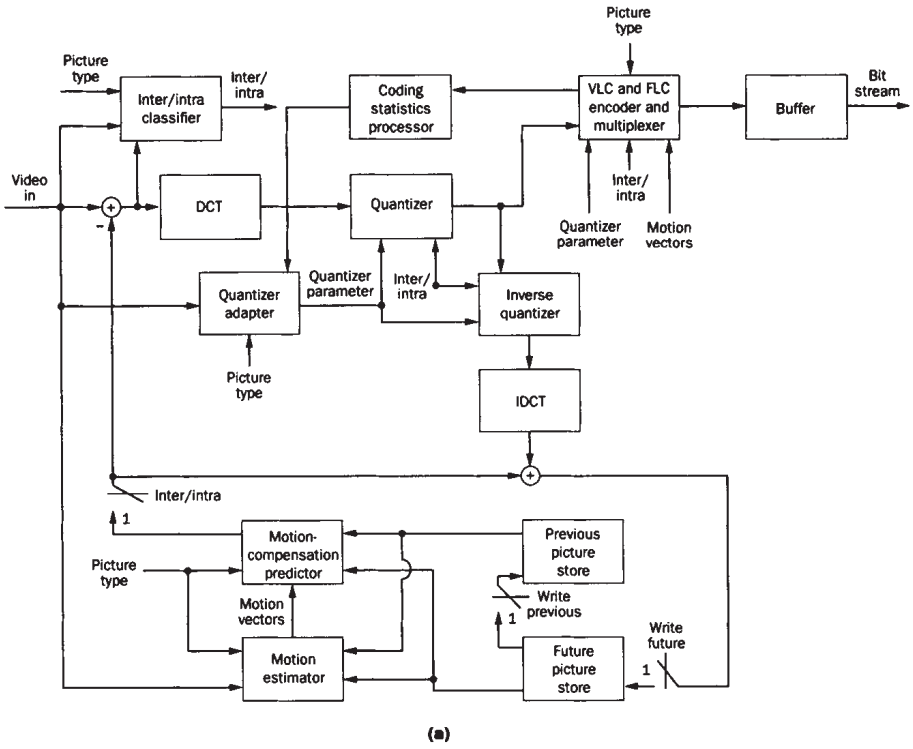


Fig. 9.7a A typical MPEG-1 encoder.

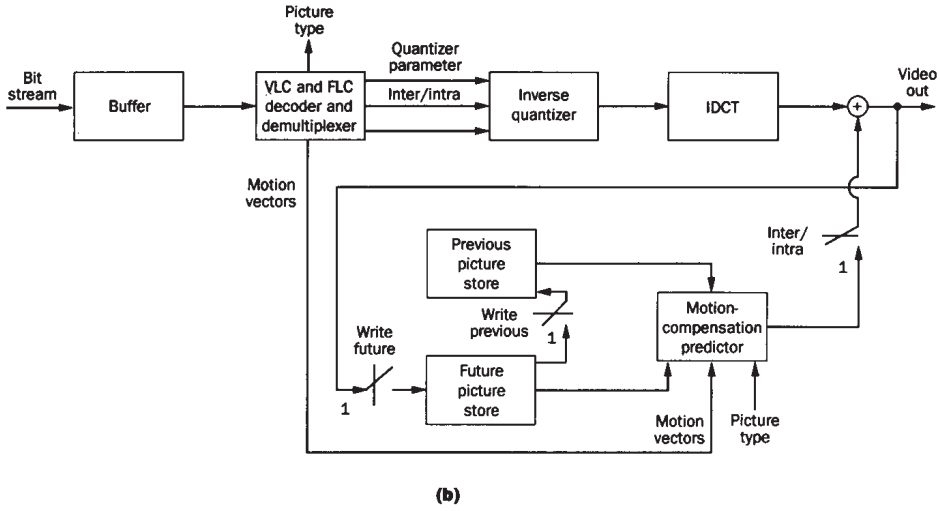


Fig. 9.7b A typical MPEG-1 decoder.

Table 9.3 Parameter Bounds for MPEG2 MP@ML Bit-Streams (K=1024)

Parameter	Upper Bound
horizontal size	720 pels
vertical size	576 lines
frame rate	30 Hz
pel rate	10.368 MHz
VBV Buffer	224 Kbytes
bit-rate	15.0 Mbits/s
color subsampling	MPEG2 4:2:0

9.2.7 Constrained Parameter Bit Streams

In order to allow manufacturers to build standard decoder chips for the most popular applications, MPEG1 has defined a set of parameter bounds for bitstreams, as well as a constrained parameters flag in the Sequence Header indicating that the bitstream conforms to those parameters. These parameter bounds are shown in Table 9.2.

9.2.8 Typical Encoder Architecture

Fig. 9.7a shows a typical MPEG1 video encoder. It is assumed that frame reordering takes place before coding, i.e., I- or P-pictures used for B-picture prediction must be coded and transmitted before any of the corresponding B-pictures, as shown in Fig. 9.3.

Input video is fed to a Motion Compensation Estimator/Predictor that feeds a prediction to the minus input of the Subtractor. For each MB, the Inter/Intra Classifier then compares the input pels with the prediction error output of the Subtractor. Typically, if the mean square prediction error exceeds the mean square pel value, an Intra MB is decided. More complicated comparisons involving DCT of both the pels and the prediction error yield somewhat better performance, but are not usually deemed worth the cost.

For Intra MBs the prediction is set to zero. Otherwise, it comes from the Predictor, as described above. The prediction error is then passed through the DCT and Quantizer before being coded, multiplexed and sent to the Buffer.

Quantized Levels are converted to reconstructed DCT coefficients by the Inverse Quantizer and then inverse transformed by the IDCT to produce a coded prediction error. The Adder adds the prediction to the prediction error and clips the result to the range 0 to 255 to produce coded pel values.

For B-pictures the Motion Compensation Estimator/Predictor uses both the Previous Picture and the Future Picture. These are kept in picture stores and remain unchanged during B-picture coding. Thus, in fact, the Inverse Quantizer, IDCT and Adder may be disabled during B-picture coding.

For I and P-pictures the coded pels output by the Adder are written to the Future Picture Store, while at the same time the old pels are copied from the Future Picture Store to the Previous Picture Store. In practice this is usually accomplished by a simple change of memory addresses.

The Coding Statistics Processor in conjunction with the Quantizer Adapter control the output bit-rate in order to conform to the Video Buffer Verifier (VBV) and to optimize the picture quality as much as possible. A simple control that works reasonably well is to define a target buffer fullness for each picture in the GOP. For each picture the *quantizer_scale* value is then adjusted periodically to try to make the actual buffer fullness meet the assigned value. More complicated controls would, in addition, exploit spatio-temporal masking in choosing the *quantizer_scale* parameter for each MB.

9.2.9 Typical Decoder Architecture

Fig. 9.7b shows a typical MPEG1 video decoder. It is basically identical to the pel reconstruction portion of the encoder. It is assumed that frame reordering takes place after decoding and video output. However, extra memory for reordering can often be avoided if during a write of the Previous Picture Store, the pels are routed also to the display.

The decoder cannot tell the size of the GOP from the bitstream parameters. Indeed it does not know until the Picture Header whether the picture is I, P or B Type. This could present problems in synchronizing audio and video were it not for the Systems part of MPEG1, which provides Time Stamps for audio, video and ancillary data. By presenting decoded information at the proper time as indicated by the Time Stamps, synchronization is assured.

Decoders often must accommodate occasional transmission bit errors. Typically, when errors are detected either through external means or internally through the arrival of illegal data, the decoder replaces the data with skipped MBs until the next Slice is detected. The visibility of errors is then rather low unless they occur in an I-picture, in which case they may be very visible throughout the GOB. A cure for this problem was developed for MPEG2 and will be described later.

9.2.10 Performance of MPEG1

In demonstrations of MPEG1 video at a bit-rate of 1.1 Mbits/s, SIF resolution pictures have been used. This resolution is roughly equivalent to one field of an interlaced NTSC or PAL picture. The quality achieved by the MPEG1 video encoder at this bit-rate has often been compared to that of VHS* videotape playback. Although the MPEG1 video standard was originally intended for operation in the neighborhood of the above bit-rate, a much wider range of resolution and bit-rates is supported by the syntax. The MPEG1 video standard thus provides a generic bit-stream syntax that can be used for a variety of applications. The MPEG1 Video Part 2 of ISO/IEC 11172 provides all the details of the syntax, complete with informative sections on encoder procedures that are outside the scope of the standard.

9.3 MPEG Second Work Item (MPEG2)

As of this writing, a second MPEG work item is nearly finished, which is colloquially called MPEG2. It is aimed at video bit-rates above 2 Mbits/s. As with MPEG1, the MPEG2 standard is written in four parts, Systems, Video,

---

* VHS, an abbreviation for Video Home System, is a registered trademark of the Victor Company of Japan.

Audio and Compliance Testing. More parts will follow. Originally, a third work item MPEG3 was planned for High Definition TV. However, the HDTV work was folded into MPEG2 at an early stage. The official name of MPEG2 is ISO/IEC 13818.

The original objective of MPEG2 was to code interlaced CCIR 601 video at a bit-rate that would serve a large number of consumer applications, and in fact one of the main differences between MPEG1 and MPEG2 is that MPEG2 handles interlace efficiently. Since the picture resolution of CCIR 601 is about four times that of the SIF of MPEG1, the bit-rate chosen for MPEG2 optimization was 4 Mbits/s. However, MPEG2 allows rates as high as 429 Gbits/s.

A bit-rate of 4 Mbits/s was deemed too low to enable high quality transmission of every CCIR 601 color sample. Thus, an MPEG2 4:2:0 format was defined to allow for 2:1 vertical subsampling of the color, in addition to the normal 2:1 horizontal color subsampling of CCIR 601. Pel positions are shown in Fig. 9.8, and as we can see, it is only slightly different than the CIF of H.261 and the SIF of MPEG1.

For interlace, the temporal integrity of the chrominance samples must be maintained. Thus, MPEG2 normally defines the first, third, etc. rows of 4:2:0 chrominance CbCr samples to be temporally the same as the first, third, etc. rows of luminance Y samples. The second, fourth, etc. rows of chrominance CbCr samples are temporally the same as the second, fourth, etc. rows of luminance Y samples. However, an override capability is available to indicate that the 4:2:0 chrominance samples are all temporally the same as the temporally first field of the frame.

At higher bit-rates the full 4:2:2 color format of CCIR 601 may be used, in which the first luminance and chrominance samples of each line are co-sited. MPEG2 also allows for a 4:4:4 color format.

9.3.1 Requirements of the MPEG2 Video Standard

The primary requirement of the MPEG2 video standard is that it should achieve the highest possible quality of the decoded video during normal play at a given bit-rate. In addition to picture quality, different applications have additional requirements even beyond those provided by MPEG1. For instance, multipoint network communications may require the ability to communicate simultaneously with SIF and CCIR 601 decoders. Communication over packet networks may require prioritization so that the network can drop low priority packets in case of congestion. Broadcasters may wish to send the same program to CCIR 601 decoders as well as to progressive scanned HDTV decoders. In order to satisfy all these requirements MPEG2 has defined a large number of capabilities.

However, not all applications will require all the features of MPEG2. Thus, to promote interoperability amongst applications, MPEG2 has designated

several sets of constrained parameters using a two-dimensional rank ordering. One of the dimensions, called *Profile* specifies the coding features supported. The other dimension, called *Level*, specifies the picture resolutions, bit-rates, etc. that can be handled. A number of Profile-Level combinations have been defined, the most important of which is called *Main Profile at Main Level*, or MP@ML for short. Parameter constraints for MP@ML are shown in Table 9.3.

9.3.2 Main Profile at Main Level - Algorithm Overview

As with MPEG1, both spatial and temporal redundancy reduction are needed for the high compression requirements of MPEG2. For progressive scanned video there is very little difference between MPEG1 and MPEG2 compression capabilities. However, interlace presents complications in removing both types of redundancy, and many features have been added to deal specifically with it. MP@ML only allows 4:2:0 color sampling.

For interlace, MPEG2 specifies a choice of two picture structures. *Field Pictures* consist of fields that are coded independently. With *Frame Pictures*, on the other hand, field pairs are combined into frames before coding. MPEG2 requires interlace to be decoded as alternate top and bottom fields.* However, either field can be temporally first within a frame.

MPEG2 allows for progressive coded pictures, but interlaced display. In fact, coding of 24 frame/s film source with 3:2 pulldown display (see Sec. 2.2.4) for 525/60 video is also supported.

9.3.2a Features of the MPEG2 Bit-Stream Syntax

The MPEG2 video standard specifies the *syntax* and *semantics* of the compressed bit stream produced by the video encoder. The standard also specifies how this bit stream is to be parsed and decoded to produce a decompressed video signal. Most of MPEG2 consists of additions to MPEG1. The Header structure of MPEG1 was maintained with many additions within each Header. For constant bit-rates, the Video Buffer Verifier (VBV) was also kept. However, unlike MPEG1, picture skipping as in H.261 is allowed.

Video Sequence Header

The *Video Sequence* Header contains basic parameters such as the size of the coded video pictures, size of the displayed video pictures, Pel Aspect Ratio (PAR), picture rate, bit-rate, VBV buffer size, Intra and NonIntra Quantizer Matrices (defaults are the same as MPEG1), Profile/Level Identification, Interlace/Progressive Display Indication, Private user data, and certain other global parameters.

---

* The top field contains the top line of the frame. The bottom field contains the second (and bottom) line of the frame.

Group of Pictures (GOP) Header

Below the Video Sequence Header is the *Group of Pictures* (GOP) Header, which provides support for random access, fast search, and editing. The GOP Header contains a time code (hours, minutes, seconds, frames) used by certain recording devices. It also contains editing flags to indicate whether the B-pictures following the first I-picture can be decoded following a random access.

Picture Header

Below the GOP is the *Picture* Header, which contains the type of picture that is present, e.g., I, P or B, as well as a *Temporal Reference* indicating the position of the picture in display order within the GOP. It also contains the parameter *vbv_delay* that indicates how long to wait after a random access before starting to decode. Without this information, a decoder buffer could underflow or overflow following a random access.

Within the Picture Header several picture coding extensions are allowed. For example, the Intra DC coefficient precision may be increased from the 8 bits of MPEG1 to as much as 10 bits. A 3:2 pulldown flag indicates, for frame pictures, that the first field of the picture should be displayed one more time following the display of the second field. An alternative scan to the DCT zigzag scan may be specified. And the presence of error concealment information may be indicated. Other information includes Picture Structure (field or frame), field temporal order (for frame pictures), progressive frame indicator, and information for reconstruction of a composite NTSC or PAL analog waveform.

Within the Picture Header a picture display extension allows for the position of a *display rectangle* to be moved on a picture by picture basis. This feature would be useful, for example, when coded pictures having IAR 16:9 are to be also received by conventional TV having IAR 4:3. This capability is also known as *Pan and Scan.*

MPEG2 Slice Header

The *Slice* Header is intended to be used for re-synchronization in the event of transmission bit errors. It is the responsibility of the encoder to choose the length of each Slice depending on the expected bit error conditions. Prediction registers used in differential encoding are reset at the start of a Slice. Each horizontal row of MBs must start with a new Slice, and the first and last MBs of a Slice cannot be skipped. In MP@ML, gaps are not allowed between Slices. The Slice Header contains the vertical position of the Slice within the picture, as well as a *quantizer_scale* parameter (corresponding to GQUANT in H.261). The Slice Header may also contain an indicator for Slices that contain only Intra MBs. These may be used in certain Fast Forward and Fast Reverse display applications.

Macroblock Header

In the *Macroblock* Header, the horizontal position (in MBs) of the first MB of each Slice is coded with the MB Address VLC. The positions of additional transmitted MBs are coded differentially with respect to the most recently transmitted MB also using the MB Address VLC. Skipped MBs are not allowed in I-pictures. In P-pictures, skipped MBs are assumed NonIntra with zero coefficients and zero motion vectors. In B-pictures, skipped MBs are assumed NonIntra with zero coefficients and motion vectors the same as the previous MB. Also included in the Macroblock Header are MB Type (Intra, NonIntra, etc.), Motion Vector Type, DCT Type, *quantizer_scale* (corresponding to MQUANT in H.261), motion vectors and coded block pattern. As with other Headers, many parameters may or may not be present, depending on MB Type.

MPEG2 has many more MB Types than MPEG1, due to the additional features provided as well as to the complexities of coding interlaced video. Some of these will be discussed below.

Block

A *Block* consists of the data for the quantized DCT coefficients of an 8×8 Block in the Macroblock. It is VLC coded as described in the next sections. MP@ML has six blocks per MB. For noncoded Blocks, the DCT coefficients are assumed to be zero.

9.3.2b Exploiting spatial redundancy

As in MPEG1, the MPEG2 video-coding algorithm employs an 8×8 Block-based two-dimensional DCT. A quantizer is applied to each DCT coefficient, which sets many of them to zero. Compression is then achieved by transmitting only the nonzero coefficients and by entropy-coding their locations and amplitudes.

The main effect of interlace in frame pictures is that since alternate scan lines come from different fields, vertical correlation is reduced when there is motion in the scene. MPEG2 provides two features for dealing with this.

First, with reduced vertical correlation, the zigzag scanning order for DCT coefficients shown in Fig. 8.8 is no longer optimum. Thus, MPEG2 has an *Alternate Scan,* shown in Fig. 9.9, that may be specified by the encoder on a picture by picture basis. A threshold on the sum of absolute vertical line differences is often sufficient for deciding when to use the alternate scan. Alternatively, DCT coefficient amplitudes after transformation and quantization could be examined.

Second, a capability for field coding within a frame picture MB is provided. That is, just prior to performing the DCT, the encoder may reorder the luminance lines within a MB so that the first 8 lines come from the top field, and the last 8 lines come from the bottom field. This reordering is undone after the IDCT in the encoder and the decoder. Chrominance blocks are not reordered in

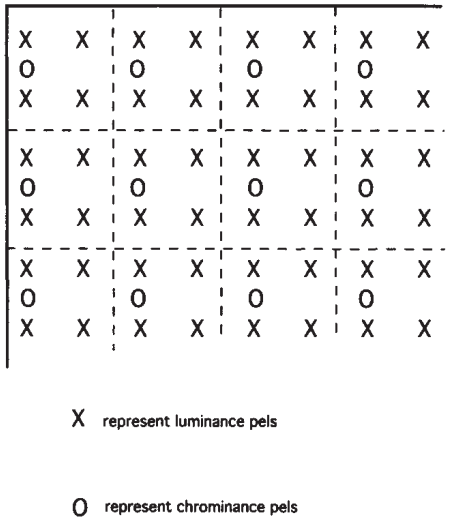


Fig. 9.8 The position of luminance and chrominance samples in MPEG2 4:2:0 format. For interlace, alternate rows of chrominance come from alternate fields.

m \ n	0	1	2	3	4	5	6	7
0	00	04	06	20	22	36	38	52
1	01	05	07	21	23	37	39	53
2	02	08	19	24	34	40	50	54
3	03	09	18	25	35	41	51	55
4	10	17	26	30	42	46	56	60
5	11	16	27	31	43	47	57	61
6	12	15	28	32	44	48	58	62
7	13	14	29	33	45	49	59	63

Fig. 9.9 MPEG2 Alternate Scan. For interlaced frame structure pictures containing motion, the alternate scan may be used instead of the zigzag scan of Fig. 8.8.

MP@ML. The effect of this reordering is to increase the vertical correlation within the luminance blocks and thus decrease the DCT coefficient energy. Again, a threshold on the sum of absolute vertical line differences is often sufficient for deciding when to use field DCT coding in a MB.

9.3.2c Exploiting temporal redundancy

As in MPEG1, the quality achieved by intraframe coding alone is not sufficient for typical MPEG2 video signals at bit-rates around 4 Mbits/s. Thus, MPEG2 uses all the temporal redundancy reduction techniques of MPEG1, plus other methods to deal with interlace.

Frame Prediction for Frame Pictures

MPEG1 exploits temporal redundancy by means of motion compensated MBs and the use of P-pictures and B-pictures. MPEG2 refers to these methods as *Frame Prediction for Frame Pictures*, and, as in MPEG1, assigns up to one motion vector to Forward Predicted Target MBs and up to two motion vectors to Bidirectionally Predicted Target MBs. Prediction MBs are taken from previously coded Reference frames, which must be either P-pictures or I-pictures. Frame Prediction works well for slow to moderate motion, as well as panning over a detailed background. Frame Prediction cannot be used in Field Pictures.

Field Prediction for Field Pictures

A second mode of prediction provided by MPEG2 for interlaced video is called *Field Prediction for Field Pictures*. This mode is conceptually similar to Frame Prediction, except that Target MB pels all come from the same field. Prediction MB pels also come from one field, which may or may not be of the same polarity (top or bottom) as the Target MB field. For P-pictures, the Prediction MBs may come from either of the two most recently coded I- or P-fields. For B-pictures, the Prediction MBs are taken from the two most recently coded I- or P-frames. Up to one motion vector is assigned to Forward Predicted Target MBs and up to two motion vectors to Bidirectionally Predicted Target MBs. This mode is not used for Frame Pictures and is useful mainly for its simplicity.

Field Prediction for Frame Pictures

A third mode of prediction for interlaced video is called *Field Prediction for Frame Pictures*. With this mode, the Target MB is first split into top-field pels and bottom-field pels. Field prediction, as defined above, is then carried out independently on each of the two parts of the Target MB. Thus, up to two motion vectors are assigned to Forward Predicted Target MBs and up to four motion vectors to Bidirectionally Predicted Target MBs. This mode is not used for Field Pictures and is useful mainly for rapid motion.

Dual Prime for P-pictures

A fourth mode of prediction, called *Dual Prime for P-pictures*, transmits one motion vector per MB. From this motion vector two preliminary predictions are computed, which are then averaged together to form the final prediction. The previous picture cannot be a B-picture for this mode. The first preliminary prediction is identical to Field Prediction, except that the Prediction pels all come from the previously coded fields having the same polarity (top or bottom) as the Target pels. Prediction pels, which are obtained using the transmitted MB motion vector, come from one field for Field Pictures and from two fields for Frame Pictures.

The second preliminary prediction is derived using computed motion vectors plus a small motion vector correction. For this prediction, Prediction pels come from the opposite polarity field as the first preliminary prediction. The computed motion vectors are obtained by a temporal scaling of the transmitted motion vector to match the field in which the Prediction pels lie. The small motion vector correction of up to one-half pel is transmitted in the MB Header.

The performance of Dual Prime prediction can rival that of B-picture prediction under some circumstances and has the advantage of lower encoding delay. However, memory bandwidth requirements are usually higher for Dual Prime.

16×8 MC for Field Pictures

A fifth mode of prediction is called 16×8 *MC for Field Pictures*. Basically, this mode splits the Field Picture MB into an upper half and lower half, and performs a separate Field Prediction for each half. Up to two motion vectors are transmitted per P-picture MB, and up to four motion vectors per B-picture MB.

It is the responsibility of the Encoder to decide which prediction mode is best. Needless to say, this can be an extremely complex process if done with full optimality. Thus, in practice numerous shortcuts are usually employed for economical implementation.

9.3.2d Quantization of DCT Coefficients

As with MPEG1, Intra coded blocks have their DC coefficients coded differentially with respect to the previous block of the same YCbCr type, unless the previous block is nonIntra or belongs to another Slice. The range of unquantized Intra DC coefficients is 0 to 8*255, which means the range of differential values is -2040 to 2040. Intra DC Differentials are quantized with a uniform mid-step quantizer with step size 8, 4 or 2, as specified in the Picture Header. Quantized Intra Differential DC Levels for MP@ML have a range of -1020 to 1020.

Intra AC coefficients are quantized with a uniform mid-step quantizer having variable step size. Unlike MPEG1, MPEG2 does not require odd representative levels. For coefficient $c[m][n]$ the IntraAC step size is given by

$$(2*quantizer\_scale*IntraQmatrix[m][n])/32, \tag{9.5}$$

where *quantizer_scale* is set by a Slice or MB Header and $IntraQmatrix[m][n]$ is the Intra quantizer matrix value for DCT frequency $[m][n]$. Again, quantization is usually performed by division and rounding.

All NonIntra coefficients are quantized in a way similar to H.261 and MPEG1, except that MPEG2 does not require odd representative levels. The NonIntra step size is given by

$$2*QUANT = (2*quantizer\_scale*NonIntraQmatrix[m][n])/32, \tag{9.6}$$

where $NonIntraQmatrix[m][n]$ is the NonIntra quantizer matrix value for DCT frequency $[m][n]$. The quantizer is nonuniform with a *dead zone* around zero, as shown in Fig. 8.11. Thus, coefficient values should be shifted toward zero by at least *QUANT* prior to dividing by $2*QUANT$ when computing quantized Levels.

Quantized Levels have a range of -2047 to 2047. In principle, this could be enforced by clipping after division by the step size. However, clipping may cause unacceptable picture distortion. A better solution is usually to increase the step size.

9.3.2e Variable Length Coding of Quantized DCT Coefficients

The Intra DC VLC is similar to MPEG1, i.e., Levels are coded as a VLC size followed by a VLI Level, where the VLC indicates the size in bits of the VLI. Quantized Intra DC Levels have a range of -1020 to 1020.

The remaining coefficients are coded either in zigzag order or alternate scan order using one of two available VLCs. Runlength-Level combinations that are not in the VLC are coded using a a 6-bit Escape, 6-bit FLC for Runlength, and a 12-bit FLC for Levels. Quantized Levels have a range of -2047 to 2047.

9.3.2f Reconstruction of Quantized DCT Coefficients

Intra DC Differential values are obtained by simple multiplication of received Levels by the specified step size. Intra AC coefficient reconstruction is simply

$$\tilde{c}[m][n] = (2*Level[m][n]*quantizer\_scale*IntraQmatrix[m][n])/32 \tag{9.7}$$

All NonIntra coefficients are reconstructed by

$$\begin{aligned}\tilde{c}[m][n] = ((&2*Level[m][n] \\ &+Sign[m][n])*quantizer\_scale*NonIntraQmatrix[m][n])/32\end{aligned} \tag{9.8}$$

All reconstructed coefficients are then clipped to the range -2048 to 2047, followed by mismatch control. Mismatch control is very different than MPEG1

or H.261. MPEG2 basically adds all the reconstructed coefficients of each block, and if the sum is even, the least significant bit of the highest frequency coefficient is changed.

9.3.2g MPEG2 Error Concealment

MPEG2 has added an important feature for concealing the effects of transmission bit-errors in MP@ML. Intra pictures may optionally contain coded motion vectors, which are used only for error concealment. Typically, a Slice that is known to be erroneous is replaced by motion compensated pels from the previous I or P-picture. Motion vectors from the slice above could be used for this process.

9.3.2h Typical Encoder/Decoder Architecture

An MPEG2 codec architecture is nearly identical in form to the MPEG1 codec of Fig. 9.7. The main difference is in the complexity of the motion estimation and the Coding Statistics Processor, which for full optimality must decide amongst many many alternatives. Most of these decisions can utilize the mean absolute prediction error of the various modes.

9.3.2i Performance of MPEG2 MP@ML

In demonstrations of MPEG2 video at a bit-rate of 4.0 Mbits/s, interlaced CCIR 601 pictures have been used, with 2:1 color subsampling both horizontally and vertically. The quality achieved by the MPEG2 video encoder is very good to excellent, depending on scene content. Error resilience has been found to be very good at bit error rates up to 10^{-4} and usable up to 10^{-3}.

9.3.3 MPEG2 Scalability Extensions

Several additions to the basic Main Profile have been defined for those applications that require increased functionality. The *Scalability* extensions basically provide for two or more separate bitstreams, or *Layers*, that can be combined to provide a single high quality video signal.

The Lower Layer or *Base Layer* bitstream can, by definition, be decoded all by itself to provide a lower quality video signal. With the incorporation of *Enhancement Layer* bitstreams, the Base Layer image quality can be increased significantly, depending on the relative bit-rates assigned to the various Layers.

Scalability is especially useful in packet transmission systems, such as Asynchronous Transfer Mode (ATM) networks, that allow for dual priority packetization. The Base Layer would be sent at high priority and the Enhancement Layer at low priority. Then in case of low priority packet loss, a reasonably good video signal could still be recovered.

9.3.3a SNR Scalability

With SNR Scalability, the Enhancement Layer provides a direct improvement in DCT coefficient accuracy by adding correction values just prior to the Base Layer decoder IDCT. Thus, the extra processing needed by the Enhancement Layer consists of variable length decoding, inverse scan, and finally, reconstruction of quantized differential coefficient values. This extension also provides for simulcast 4:2:2 color information to be coded in the Enhancement Layer, with 4:2:0 color in the Base Layer.

SNR Scalability might be used in video distribution networks, where the full enhanced video would be required for editing and overlays prior to broadcast, whereas the Base Layer by itself would be sufficient for broadcast of nonedited video.

9.3.3b Data Partitioning

Data Partitioning is basically a simplified version of SNR Scalability. With this extension, up to a certain number (specified in the Slice Header) of low frequency run-level codewords in a block are sent in the Base Layer. The remaining high frequency codewords, if any, are sent in the Enhancement Layer.

The main attraction of Data Partitioning is its simplicity and reduced overhead compared with other scalability extensions. The only extra processing needed is a codeword counter to control switching from one layer to the other. Because of its reduced overhead, the enhanced image quality of Data Partitioning is typically very close to that of single layer coding.

9.3.3c Spatial Scalability

With Spatial Scalability, the Base Layer video has a lower spatial resolution than the original video. The Enhancement Layer is then combined with the Base Layer to produce a decoded image with the original resolution. This feature can be used for interworking between video standards, e.g., SIF Base Layer and CCIR 601 original, or CCIR 601 Base Layer and HDTV original.

Simple Pyramid Coding can be used to implement Spatial Scalability, and MPEG2 allows for this. First, the resolution of the decoded Base Layer is increased to that of the Enhancement Layer by interpolation and filtering. Then the difference between the original video and the upsampled Base Layer is coded in the Enhancement Layer.

However, MPEG2 also provides for a much more powerful spatio-temporal prediction process than simple pyramid coding. In coding the Enhancement Layer MBs, the predictor has three choices. It can use the pyramid coding spatial prediction described above, or it can use the normal MPEG2 motion compensated temporal prediction, or it can use a weighted average of the two. The result is a considerable improvement in coding efficiency.

9.3.3d Temporal Scalability

With Temporal Scalability, the Base Layer video has a lower temporal resolution, i.e., frame rate, than the original video. The Enhancement Layer is then combined with the Base Layer to produce a decoded image with the original frame rate. For example this feature can be used for interworking between an interlaced CCIR 601 Base Layer and progressive scanned original video at 50 or 60 Hz frame rate.

Temporal Scalability also uses a powerful temporal prediction process for encoding the Enhancement Layer. The motion compensated prediction can be taken from adjacent Base Layer pictures as well as adjacent Enhancement Layer pictures.

9.3.4 Other Profiles and Levels

In addition to the Main Profile, several other MPEG2 Profiles have been defined as of this writing. The *Simple* Profile is basically the Main Profile without B-pictures. The *SNR Scalable* Profile adds SNR Scalability to the main Profile. The *Spatially Scalable* Profile adds Spatial Scalability to the SNR Scalable Profile, and the *High* Profile adds 4:2:2 color to the Spatially Scalable Profile. All scalable Profiles are limited to at most three layers.

As stated previously, the Main Level is defined basically for CCIR 601 video. In addition, the *Simple* Level is defined for SIF video. Two higher levels for HDTV are the *High-1440* Level, with a maximum of 1440 pels/line, and the *High* Level, with a maximum of 1920 pels/line.

Not all Profile/Level combinations have been standardized. In the future, additional Profiles and Levels will be defined for various applications, as demand dictates.

References

9.1 ISO/IEC 11172, Information Technology-Coding of moving pictures and associated audio for digital storage media up to about 1.5 Mbit/s

9.2 D. J. LeGall, "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of the ACM*, Vol. 34, No.4, April 1991, pp. 47-58.

9.3 "Digital Video," *IEEE Spectrum*, Vol. 29, No.3, March 1992.

9.4 ISO/IEC 13818, Information Technology - Generic coding of moving pictures and associated audio information.

Questions for Understanding

9.1 Describe the two SIF video formats? How do they differ from CIF?

9.2 How does JPEG coding differ from that of MPEG1 I-pictures?

9.3 How does P*64 coding differ from that of MPEG1 P-pictures?

9.4 What are advantages and disadvantages of MPEG1 B-Pictures?

9.5 How does MPEG support random access in stored bit streams.

9.6 How are MPEG1 Slices different than P*64 GOBs?

9.7 If the first five quantized DCT coefficients of an MPEG1 Inter block have levels 20, 0, 18, -3 and 22, what code words are used to code them? What are the code words if the block is the first Intra block of the Slice? Assume no quantizer matrices were transmitted.

9.8 If QUANT = 4, what are the reconstructed coefficient values for the above example?

9.9 If QUANT = 5, what are the reconstructed coefficient values for the above example?

9.10 Repeat the above three examples for MPEG2.

9.11 What are the MPEG2 picture structures?

9.12 How is the MPEG2 3:2 pulldown flag used?

9.13 Describe the five MPEG2 temporal prediction modes.

9.14 Describe the four MPEG2 Scalability extensions.

10

High Definition Television

10.1 Introduction

Television was initially intended for sports, news and entertainment applications and was almost entirely distributed by terrestrial broadcasts. In North America, since the adoption of the NTSC standard in 1941, only two significant modifications have been made to the television signals, namely the addition of color in 1953 and stereo sound in 1984. Both of these modifications were made in an backward compatible manner, i.e., by modifying the transmitted signal in such a way that existing receivers could still decode the new signal, while newer receivers have enhanced functionality due to the signal modifications. Television is now finding applications in telecommunications (e.g. videoconferencing), computing (e.g. desktop video), medicine (e.g. telemedicine) and education (e.g. distance learning). At the same time, television is being distributed by other means, such as cable (fiber or coaxial), video tapes or disks (analog or digital) and satellites (e.g. direct broadcast). Moreover, the television signal packaged for terrestrial broadcasting is very wasteful of the broadcast spectrum and can benefit from modern signal processing. In addition, the average size of television sets has grown steadily, and the average viewer is getting closer to video displays, particularly in applications other than transmitted entertainment television. These newer applications, delivery media and viewing habits are exposing various limitations of the television standard established over five decades ago. High definition television systems now being designed are expected to be more suitable for these newer applications and to interoperate easily with the evolving telecommunications and computing infra-structure. HDTV will be an entirely new high-resolution and wide-screen television system producing much better quality pictures and sound, thereby creating a higher sensation of reality that far exceeds that of the current television systems. In this chapter, we describe the

main characteristics of HDTV, the current state in Europe, Japan, and the United States, and the expected future evolution.

10.2 What is HDTV?

There have been a number of proposals for HDTV, and there is only partial agreement on the definition of HDTV at this time. However, there are a number of commonly agreed aspects that will be described in this section. HDTV is characterized by increased spatial and temporal resolution; increased image aspect ratio (i.e., wider image); multichannel CD-quality surround sound; reduced artifacts compared with current composite analog television; bandwidth compression and channel encoding to make better utilization of the terrestrial spectrum; and finally, better interoperability with the evolving telecommunications and computing infrastructure. We will look at each of these aspects in this chapter.

First and foremost, HDTV, as produced in a studio and received in consumer homes, must create a much better picture. This is done by increasing the spatial resolution by approximately a factor of two in both the horizontal and vertical dimensions. Thus, we get a picture that has about 1000 scan lines and more than 1000 pels per scan line. In addition, it is widely accepted (but not fully agreed to) that progressive (noninterlace) scanning at about 60 frames/sec will render better fast-action sports, while also eliminating the artifacts associated with interlace.* The result of this is that the number of active pels in an HDTV signal increases by about a factor of five, with a corresponding increase in the analog bandwidth. From consumers' experience in cinema, a wider image aspect ratio is also preferred. Most HDTV systems specify the image aspect ratio to be 16:9.

Improvement in audio is also crucial to HDTV. This means multi (4 to 6) channel surround sound, in which each channel is independently transmitted. In digital HDTV, each audio channel is independently compressed, and these compressed bits are multiplexed with compressed bits from video, plus bits for closed captioning, teletext, encryption, addressing, program identification and other data services in a layered fashion. Audio fidelity is nearly as good as that obtainable from the current generation of CDs.

---

* Proponents of interlace argue that interlace represents a better trade-off between spatial and temporal resolution (i.e., for a manageably low overall pel-rate, interlace allows 1000 lines), interlace-equipment exists (e.g. 1125/60 based on the SMPTE 240 system) and is cheaper. Therefore, they argue, one should start with an interlaced signal and migrate to progressive scan at a later time.

10.3 History of HDTV

In this section we give a brief history of the development of HDTV in Japan, Europe and the United States.

HDTV in Japan

In the 1970's, the Japan Broadcasting Corporation[10.1] (NHK) began to design the next generation television system. The main systems work was done by NHK, but the substantial infra-structure equipment required for a complete system (e.g. cameras, recorders and displays) was developed by other Japanese companies. The NHK developed system had 1125, 2:1 interlaced scan lines and frequency multiplexing of color (but not in the same bandwidth as the luminance). The initial system was intended for a wideband (100 MHz) satellite channel to supplement (not replace) NTSC. Further developments led to the MUSE* compression system that would fit into a normal 27 MHz satellite transponder.[10.2] In the mid 80's, it was realized that the production system could be distinct from the transmission system. Since then, the NHK 1125/60 system has been proposed as a production and international exchange standard, on the basis that much of the equipment has already been developed. However, there is still considerable disagreement in selection of the scanning parameters of the production standard. Meanwhile, satellite HDTV has been in service in Japan for several years, and the receiver costs have come down steadily (but still remain very high). MUSE was further modified to fit into the 6MHz terrestrial channel (called Narrow-MUSE) in order to compete in the FCC-sponsored competition in the United States. Even though the NHK standard itself remains controversial, it gave rise to substantial developments in much of the equipment required for both professional and consumer HDTV. More recently, Japanese efforts are being revived to make HDTV digital using both compression and digital modulation for channel coding.

European HDTV

Although the European Broadcasting Union (EBU), had given some indication of support[10.3] for the NHK-1125/60 system for professional studio use, in 1986 the broadcasting and manufacturing interests in Europe agreed on their own high definition standards, called HD-MAC.† HD-MAC was expected to be delivered exclusively by direct broadcast satellite (DBS), and it was configured to be backward compatible with the newer versions of PAL and SECAM (i.e., D-MAC). Thus, HD-MAC would supplement PAL and SECAM

---

* MUSE = Multiple Sub-Nyquist sampling encoding.

† HD-MAC = High Definition – Multiplexed Analogue Components.

and, therefore, would never result in retiring them. Significant expenditures were made by government and industry to develop HD-MAC under project Eureka-95. Nevertheless, the future of HD-MAC appears bleak. Although it has not been officially withdrawn, sentiment in Europe is now leaning toward a system using digital compression as well as digital modulation for transmission. There are many opportunities to select a standard, or major elements of a standard, such that interoperability within Europe as well as with Japan and the U.S. could be facilitated.

HDTV in North America

Since television is delivered over many media, the service providers for each medium were concerned that HDTV might succeed over an alternative medium, thus taking their audience away. Broadcasters, under pressure from the alternative media (since HDTV may not increase an already saturated audience's total viewing hours), petitioned the FCC (Federal Communications Commission of the United States) in 1987 to assess the effect of HDTV on the existing broadcasting structure. Broadcasters were also feeling the pressure from the mobile telecommunications service providers who wanted the unused (taboo channel) spectrum for wireless services. Before the petition, in spite of pressure from several industry groups such as SMPTE 240M (See Chapter 2), a slightly modified version of the NHK 1125/60 system was rejected as the American National Standard (ANSI) for production. In September 1987, the FCC established the Advisory Committee on Advanced Television Service (ACATS) to make a recommendation on advanced television (including HDTV). ACATS proposed hardware testing the best of the 23 proposed systems then vying to be the standard and set up a process to conduct such tests. As the testing drew closer, only six systems remained. Compared to the previously proposed systems, four out of these six systems emphasized simulcasting in the taboo channels, digital compression and transmission. When the testing concluded in late 1992, all four digital systems remained in the running, but a clear winner could not be chosen. In May 1993, the remaining competitors agreed to pool their knowledge to create a combined "best of the best" system called the Grand Alliance HDTV (GA-HDTV) system.

10.4 Problems of Current Analog TV

Current composite analog television has many defects and artifacts. Some are inherent and some are due to the addition of color on a subcarrier without increasing the overall bandwidth. HDTV attempts to alleviate many of these artifacts. For example, **inter-line flicker, line crawl and vertical aliasing** are largely a result of interlaced scanning. These effects are more pronounced in pictures containing slowly moving high-resolution graphics and text with sharp edges. The computer industry has largely abandoned interlace scanning in order

to avoid these problems. Progressive scanning and transmission of HDTV will eliminate these effects almost entirely.

Large area flicker as seen from a close distance or on large screens is due to our ability to resolve temporal brightness variations at frequencies even beyond 60Hz when the peripheral areas of vision are engaged. The cost of overcoming this problem in terms of bandwidth (i.e., by increasing the frame rate) is too large and is, therefore, not addressed in any current HDTV proposal. Computer displays employ much higher frame rates (up to 72 Hz) to overcome large area flicker.

Static raster visibility is more apparent on larger displays since viewers can make out individual scan lines. Increased spatial resolution in HDTV will reduce this effect substantially.

Finally, **cross-color and cross-luminance** are a result of color modulation used to create the composite signal. HDTV will use component signals. Moreover, in digital HDTV, each color component will be compressed separately, with the compressed bits multiplexed in time. Thus, these two effects will be eliminated entirely.

With the present analog television, a number of other artifacts also arise, due to transmission. For example, **ghosts**, which are due to multiple receptions of the same signal, cannot be entirely removed from the analog signal. In digital HDTV, ghost cancellation can be done more successfully, generally leaving no impact on the transmitted bit stream.

Another transmission problem is interference from other TV signals. As an example, the NTSC signal, containing high energy and sync pulses with sharp transitions, has poor spectrum utilization due to a high degree of **co-channel*** and **adjacent channel** interference. Co-channel interference constrains the minimum geographic distance between transmitters on the same frequency band. Adjacent channel interference precludes using spectrally adjacent channels to serve the same area unless the transmitters are located together, thus assuring near-equivalent, non-interfering signal strengths at all receiving locations. Both of these interferences are reduced by keeping adjacent channels in the spectrum vacant (i.e. taboo) at any one place, while using these vacant channels at other surrounding places. As a result, in the United States only about 20 channels are usable in each location out of a total of 68. In the United Kingdom, only about 4 out of 44 channels are usable.

10.5 Advantages of Digital HDTV

Unfortunately, increasing the spatial and temporal resolution of the HDTV signal also increases its analog bandwidth. In the currently allocated terrestrial

---

* Two signals from different places, but in the same frequency band.

broadcast spectrum, such an analog signal could not be accommodated in any location. Therefore, all the newer HDTV proposals employ digital bandwidth compression. Digital HDTV reduces the bit-rate from approximately 1 Gbits/s to about 20Mbits/s. This digital signal is incompatible with the current television system and, therefore, cannot be decoded by today's television sets.

Such a compression, by itself, however, is not enough. In order to improve spectrum utilization, it is also necessary to reduce interference to and from other broadcast signals, including the analog TV channels.

Digital HDTV modulates the compressed audio and video bits into a signal that has much lower power, while requiring the same analog bandwidth as the current analog television. In addition, most HDTV proposals simulcast the HDTV signal along with the current analog television signal having the same content. Simulcasting is shown in Figs. 10.1 and 10.2. A low power, modulated digital HDTV signal will be transmitted in a taboo channel that is currently unused because of co-channel and adjacent channel interference considerations. Simultaneously, in an existing channel, the same content (down converted to current television resolution) will be transmitted and received by today's analog television sets. By proper shaping of the modulated HDTV signal, it is possible to avoid its interference into distant co-channel as well as nearby adjacent channel analog television signals.

At the same time, distant co-channel and adjacent channel analog television signals must not degrade the HDTV pictures significantly. This allows previously unused terrestrial spectrum to be used for HDTV as well as digital conventional TV. Consumers with current television sets will receive television as they do now. However, consumers willing to spend an additional amount will be able to receive HDTV in currently unused channels. As the HDTV service takes hold in the market place, spectrally inefficient analog television can be retired to release additional spectrum for additional HDTV channels or other services.*

Another desirable feature for HDTV systems is easy interoperability with the evolving telecommunications and computing infra-structure. Such interoperability will allow easy introduction of different services and applications, and at the same time reduce costs due to commonality of components across different applications. There are several characteristics that improve interoperability. Progressive scanning, square pels and digital representation improve interoperability with computers. In addition, they facilitate signal processing required to convert HDTV signals to other formats either for editing, storage, special effects, or down-conversions to current

---

* In the United States, the FCC plans to retire NTSC after 15 years from the start of the HDTV broadcasts.

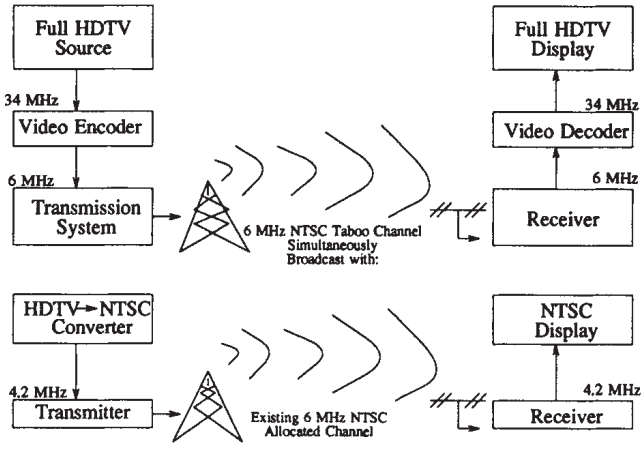


Fig. 10.1 Simulcasting of Digital HDTV with NTSC. Under current channel assignments, many channels remain vacant (i.e. taboo) to avoid co-channel and adjacent channel interference. Since digital signals have lower transmission power, they can use the taboo channels in the new channel assignment.

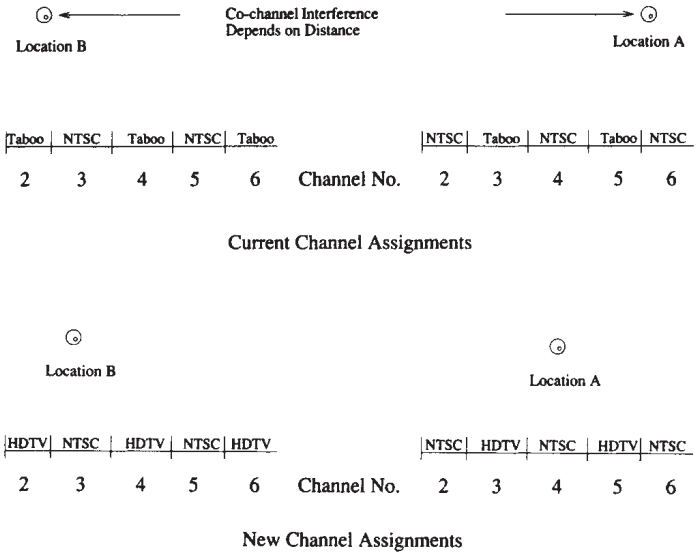


Fig. 10.2 HDTV Channel Assignments. This picture is somewhat simplified. In each geographical area, one needs to examine the channel assignments based on co-channel and adjacent channel interference between current TV and HDTV.

television. The use of a standard compression algorithm (e.g. MPEG) transforms raw, high-definition audio and video into a coded bit stream, which is nothing more than a set of computer instructions and data that can be executed by the receiver to create sound and pictures.

Further improvement in interoperability at the transport layer is achieved by encapsulating audio and video bit-streams into fixed-size packets. This facilitates the use of forward error correction (necessary to overcome transmission errors) and provides synchronization and extensibility by the use of packet identifiers, headers and descriptors. In addition, it allows easy conversion of HDTV packets into other constant size packets adopted by the telecommunications industry, e.g., the Asynchronous Transfer Mode (ATM) standard for data transport.

Fitting the HDTV signal into an existing television channel (e.g. 6MHz for NTSC) improves interoperability with existing terrestial broadcasting, with cable TV, and with certain kinds of video-tape recorder technology in which the modulated digital HDTV signal is stored as an analog signal.*

10.6 GA-HDTV System[10.4]

The Grand Alliance HDTV system is a proposed transmission standard considered by the FCC for North America. It is not a production standard, display standard or a consumer interface standard. It is expected that a production standard that is optimized for operation within a studio, but at the same time interoperable with the transmission standard, will emerge once the transmission standard is approved. Efforts to this end are underway within the Advanced Television Systems Committee (ATSC). Similarly using the transmitted signal, different manufacturers of consumer television receivers will create optimum displays consistent with picture quality obtainable for a given cost.

The GA-HDTV supports signals in multiple formats as follows:

1280 pels × 720 lines at frame rates of 24, 30, 60 Hz progressive

and

1920 pels × 1080 lines at frame rates of 24Hz, 30Hz progressive and 60Hz interlaced.†

Frame rates may also be divided by 1.001 in systems that require compatibility with NTSC. With a 16:9 image aspect ratio these formats create square pels for

---

* Current VCR's might be modified without changing the basic analog technology, but existing recorders would not be directly interoperable

† Also under consideration is 1440 pels, 60 Hz interlaced, for bit rates that cannot support 1920 pels.

both 720 and 1080 line rasters. Such flexibility in scanning allows different industries, program producers, application developers and users to optimize among resolution, frame rate, compression and interlace artifacts.

The Grand Alliance would have wished to include a 1920 × 1080 progressive scan format at 60Hz. However, the 6MHz terrestrial broadcast channel does not allow for the required bit rate. As the NTSC audience declines and the interference into NTSC becomes less of a concern, there will be an opportunity to increase the data rate to support, in a compatible manner, 1920 × 1080 progressive scan at a 60Hz frame rate. The GA-HDTV system may allow for temporal scalability, so that an enhanced bit stream to handle 1920 × 1080 progressive scan at a 60Hz frame rate can be created in the future.

The GA-HDTV system incorporates a subset of the MPEG-2 compression standard described in Chapter 9. In addition, it includes several enhancements in the encoder built for initial FCC testing, e.g., the GA-HDTV system can process film originated material at 24 or 30 frames/s by adaptively adjusting the compression algorithm at the encoder. Also, compression efficiency is improved by appropriate prefiltering/subsampling to handle signals of different formats and of varying quality (e.g. noisy signals). Among the compression techniques used are: DCT of motion compensated signals with field/frame prediction and transform, flexible refreshing (both I-frame as well as pseudo-random intra-blocks), bidirectional prediction (B-frames) and forward analysis/perceptual selection for rate control and best picture quality.

The GA-HDTV system uses five-channel audio with surround sound, which compresses into a 384 kbits/s bit stream.* Audio, video and auxiliary data are packaged into fixed-length packets in conformity with the MPEG2 Systems transport layer. MPEG2 transport packets are 188 bytes long, consisting of up to 184 bytes of payload and 4 bytes of header. These packets easily interoperate with ATM since they have approximately a 4:1 ratio with ATM packets, which are 53 bytes long with 5 bytes of header.

The modulation scheme chosen for over-the-air is 8-VSB, which gives 32.28 Mbits/s of raw data in a 6 MHz bandwidth. After error correction, 19.3 Mbits/s are available. Coaxial cable can carry more than this in a 6 Mhz channel, since there is no co-channel and very little adjacent channel interference. Cable standards are under discussion.

Thus, the GA-HDTV system attempts to make the best compromise to suit different applications and provide for maximum interoperability.

---

* This is composed of five full bandwidth and one lower bandwidth audio channels.

10.7 Future of HDTV

HDTV faces significant challenges to its widespread deployment. Many people believe that the current resolution of television is adequate for most applications and that television should evolve by being more interactive. Some believe that better use of bandwidth could be made by filling it with a larger number of lower resolution programs and then interactively choosing between them. For example, using 20 Mbits/s, which is possible in the 6MHz bandwidth, one can accommodate 4 to 10 compressed digital 525-line NTSC programs. Viewers could then select between them with ease, provided there is a good user interface. Many people also believe that the precious spectrum should not be used to transmit television to fixed locations. Larger and larger fractions of modern industrialized countries are installing fiber-optic cables having enormous traffic capability. Television could reach many people through such fixed wired connections. Some argue that terrestrial spectrum should be reserved for mobile services, which are in great demand as a result of cost reductions due to low power, compact electronics and better batteries.

The high cost of HDTV sets may be another inhibiting factor, although with growing demand, both the electronics and displays should become cheaper over time. ACATS and the Grand Alliance companies in the United States have done much to quantify the costs to broadcasters of implementing HDTV. While the costs look more manageable than ever before, broadcasters are still unenthusiastic about the business value of HDTV. However, they are concerned that other media (tape, DBS, cable) might deliver HDTV to the public before them or that the FCC might give the taboo channel spectrum to other services if the broadcasters do not use it.

HDTV may also grow by other means. In the United States, the national information infrastructure (NII) is being created. Digital HDTV, with its flexible data transport structure, is an immediate opportunity to deploy an important part of the interconnected web of information super highways that make up the NII. GA-HDTV's interoperability with computing and telecommunication platforms may allow many uses of HDTV for medicine, information access, and education. Thus, it is clear that many of the technical impediments to HDTV are being removed. Customers and the market place will decide which way it will evolve.

References

10.1 T. Fujio, "High-Definition Television System", Proceedings of IEEE vol. 73, No. 4, April 1985, pp. 647-655.

10.2 Y. Ninoniya, et. al., "HDTV Broadcasting and Transmission Systems - MUSE", Proc. of 3rd Int'l. Colloq. on Advanced TV Systems, Ottawa, Canada, Oct 1987, pp. 4.1.1 - 4.1.31.

10.3 D. I. Crawford, "High Definition Television - Parameters and Transmission", British Telecom Technology Journal, Vol. 5, No. 4, Oct. 1987, pp. 76-83.

10.4 Grand Alliance High Definition Television System, GA-HDTV Document, Nov. 1993.

Questions for Understanding

10.1 What are the pros and cons of interlaced and progressive video formats?

10.2 What are the pros and cons of digital transmission of HDTV?

10.3 What are taboo channels? Why are they needed?

10.4 How is interoperability enhanced between HDTV, telecommunications and computers?

10.5 Which video formats are defined by the GA-HDTV system?

11

Postscript

In order to transmit video information at the minimum bit rate for a given quality of reproduction it is necessary to exploit our understanding of many branches of science. Ideally one should have an appreciation of vision, signal theory, display devices, and so on. As engineers we are concerned with complex stimuli, their perception, as well as the final utilization of the perceived information. Knowledge of these is often unavailable or sketchy, forcing us to design encoders based on a relatively primitive understanding of the problem. The ultimate limits of bit-rate compression will only be approached, we believe, as our knowledge of stimuli, perception and utilization increases.

Thus, in a discussion on the directions and limitations of video bit-rate compression we are very aware of *our* limitations. In this book our modest objective of defining the state-of-the-art is, we are well aware, open to the criticisms of over-simplification, serious omissions and factual disagreement. As for where the subject is heading and its inherent limitations, we confess myopia and will not be surprised by a discovery that could not have been extrapolated from existing thinking and known ignorances.

The conventional representation of a digital communication link for the transmission of pictorial information is shown in Fig. 11.1. The function of the source encoder is to operate on an analog signal $x(t)$, and to convert it into a stream of binary digits $s(t)$. The source decoder at the receiver accepts a binary signal $S(t)$ and produces a continuous signal $X(t)$. It may not be necessary to ensure $X(t) \simeq x(t)$, but what does matter is that after transduction, $X(t)$ should be perceived as $x(t)$ subject to an acceptable quality criterion. Although $x(t)$ does not always have to be identical to $X(t)$, system engineers do prefer that $s(t) = S(t)$, i.e., the channel appears ideal. Most practical channels contain imperfections that are largely overcome by pre-processing and post-processing of the binary signals $s(t)$ and $S(t)$ by the channel codec and terminal equipment.

The purpose of this chapter is to discuss limitations of source encoding. However, Fig. 11.1 demonstrates that $X(t)$ is also dependent on the channel terminal equipment, the channel codec and, of course, the channel. Thus, encoding picture signals is not merely a source encoding problem, but engulfs the complete communication system. For example, if the channel is known to result in a high bit-error-rate (ber) then the effect on the recovered signal $X(t)$ may be mitigated by altering the modulation and regeneration strategies, increasing the length of the check bits in the channel coding words, altering the source encoding algorithm, or combinations of all of these. The conventional arrangement of source and channel codecs may be altered, even merged. Post processing of $X(t)$ can also be successfully employed.

However, in this short discourse we confine ourselves to the limitations of the source codec, assuming the channel codec has been designed to ensure that $s(t) = S(t)$. We thereby ignore the limitations imposed by the channel.

11.1 Signal Sources

Video processing or transmission systems typically start with a two-dimensional distribution of light intensity. Thus, three dimensional scenes must first be projected onto a two dimensional plane by an optical imaging system. Color pictures can usually be represented by such light intensity distributions in three primary bands of wavelengths. If moving objects are to be accommodated, all three light intensities change with time.

Television raster scanning produces 25 to 30 frames per second with 2-to-1 interlacing. Each frame is composed of many lines, e.g. 525 (Japan, North America), 625 (elsewhere), ≈280 for videotelephone. Over 2,000 lines are often used for documents and photographs. Television is bandlimited between dc and 4 or 6 MHz, with each scan line having frequency components up to about 250 cycles. Photographs and documents typically require 500 to 2,000 cycles per scan line, the width of the item being the determining factor. In color pictures, chrominance bandwidth is typically about one-fourth the luminance bandwidth.

Video is characterized by high signal energy in low to mid-band frequencies, and low energies at high frequencies. In raster scanned video, energy peaks occur at multiples of the line scan rate, with significant gaps between the peaks. Each of the line-rate harmonic peaks is made up of many sub-peaks separated by the field rate. Color multiplexing techniques often use these gaps to convey color information in the composite waveform.

Video statistics are non-stationary even over the short term. Long-term autocorrelation functions for raster scanned pictures have periodic components that depend on the scan and field rates. Two dimensional picture autocorrelation functions typically decrease monotonically with respect to spatial distance. The probability density function (PDF) of color component signals is too non-stationary to be quantified, although component differential signals tend to be Laplacian. PDF's are often used in quantizer design to minimize the mean

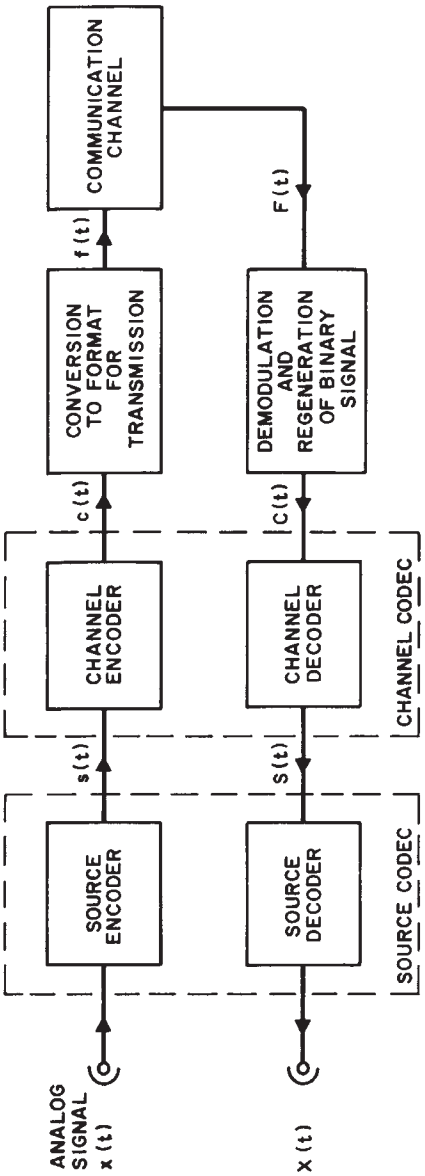


Fig. 11.1 Basic transmission system.

square error. However, many workers in the field (including the authors) reject this criterion, preferring instead criteria based on human perception. Activity factors may be useful in video, e.g., low-detail/high-detail areas, stationary/moving areas. The frequency spectra are related to both the activity factors and autocorrelation functions.

Pictures tend to have many local areas of similar color, and boundaries of objects are, more or less, continuous. In television, luminance levels in most areas do not change by large amounts between frames. Black/white graphics have large areas of constant luminance.

Such predictable structure in the video waveforms is termed *statistical redundancy*. Given the short term past, it enables the prediction, with reasonable accuracy, of the short term future. Removal of this redundancy can greatly reduce the bit-rate required for representing the video information.

11.2 The Human Receiver

The eye is the organ of sight, having at its rear an inner nervous coating known as the retina. Rays of light pass through the cornea, aqueous humour, lens and vitreous body to form an image on the retina. The central area of the retina, known as the fovea, provides high resolution and good color vision in about 1° of solid angle. The images on the two retinas are sent along the two optic nerves, one for each eye, until they meet at the optic chiasma, where half the fibers of each nerve diverge to opposite sides of the brain. This enables stereo vision and perception.

The eye behaves as a two-dimensional low-pass filter for spatial patterns, with a high-frequency cut-off of about 60 cycles per degree of foveal vision and significant low-frequency attenuation below about 0.5 cycles. Thus, high spatial frequencies in the image are not seen and need not be transmitted. The eye also acts as a temporal band-pass filter having a high-frequency cut-off between 50 and 70 Hz depending on viewing conditions. Flicker is more disturbing at high display luminances and low spatial frequencies.

Noise and amplitude distortion are generally less visible at high luminance levels than at mid and low luminance values, again depending on viewing conditions such as overall scene brightness and ambient room lighting. High and low frequency noise is less visible than mid-frequency noise. Amplitude distortions are also less visible near color transitions, such as occur at boundaries of objects in a scene. This is termed *spatial masking*, since the transitions *mask* the distortions. Although amplitude distortions are less visible near color transitions, small spatial shifts in the transitions themselves are easily visible and annoying.

Temporal masking also occurs. For example, shortly after a television scene change, the viewer is relatively insensitive to distortion and loss of resolution. This is also true of objects in a scene that are moving in an erratic and unpredictable fashion. However, if a viewer is able to track a moving

object, then resolution and distortion requirements are the same as for stationary areas of a picture.

As the variety of encoding algorithms increases so do the types of resulting degradations. If perception of these degradations were thoroughly understood, the quality of reproduction of a particular image encoding strategy could be ascertained by objective measurements of signal parameters. The current situation is one of ad-hoc objective measurements,[11.1] each trying to relate subjective observations with each new encoding algorithm or each type of degradation. Old methods of signal-to-noise ratio, spectral distance measures, pulse shapes etc., are frequently inadequate. To postulate a new objective measure, subjective testing must be done. Here, tests are made on a small sample of the population, and by statistical methods the effect on the entire population is estimated, i.e. a *statistical eye* is determined. Subjective testing is controversial. Should simple grading, bad to excellent in 5 steps, or multi-dimensional analysis be used? What form should the test take, e.g., type of picture detail, amount of motion, etc.? However, what is even more in dispute is relating subjective testing results to objective measurements. Our inability to do this, implicit in our ignorance of perception, is a serious impediment both to communication between research scientists and to source encoding itself. Only when perception is properly understood will we have accurate objective measures. However the day when we can, with confidence, objectively evaluate a *new* impairment without recourse to subjective testing seems very remote.

11.3 Waveform Encoding

In waveform coding[11.2,11.3], a continuous *analog* signal $x(t)$ is encoded into a stream of bits by the source encoder, and, from these bits, a decoder recovers a signal $X(t)$. The design objective in waveform encoding is that, for a given bit-rate, $X(t)$ should be as close a replica of $x(t)$ as possible. Since many $x(t)$'s can produce the same $X(t)$ the difference $n(t) = x(t) - X(t)$ cannot, in general, be zero all the time. This *quantization noise* is a fundamental limitation of finite bit-rate coding. For example, in pulse code modulation (PCM), $x(t)$ is sampled at the Nyquist rate, and each sample is represented by a binary number, i.e., quantized. The decoder converts the binary numbers back to analog samples and low-pass filters them to obtain $X(t)$. The bit-rate is the product of the sampling rate and the binary word length of each sample, the latter fixing the accuracy of conversion.

The signals from most TV cameras are already companded, i.e., a compressed nonlinear function of scene luminance*. Eight-bit uniform quantization of this companded signal gives imperceptible quantization noise in most cases. Single pictures, e.g., photographs, typically require one bit less quantization accuracy than television, where the quantization noise is time

---

* Henceforth, we will consider mainly luminance signal encoding since it consumes most of the channel capacity. The smaller bandwidth chrominance signal can be encoded similarly.

varying and, therefore, much more visible. For black/white images only one-bit quantization is required.

If perceptible quantization noise can be tolerated, then coarser quantization can be used and the bit rate reduced. The addition of random, or pseudo random noise prior to quantization, called dithering, changes the quantization error from being conditional on the input signal to approximately white noise and gives improved subjective results.

Successive pels of video are often highly correlated. In addition, periodicities exist in the waveform and lead to high correlations between pels that are separated, in some cases, by many sampling epochs, e.g., line, field or frame period. Predictive coding[11.2] (also called differential PCM or DPCM) exploits these correlations by using previously transmitted quantized pels to form a prediction of the current pel to be encoded. The difference between the actual pel value and its prediction is quantized, binary encoded and transmitted. The decoder is able to form the same prediction as the encoder (in the absence of transmission errors) because it has access to the same quantized pels. By adding the received quantized differential signal to the prediction the decoded quantized pel is obtained, and the signal is recovered by low pass filtering.

The predictor may be a linear, nonlinear or an adaptive function of previously encoded samples. Similarly the quantizer may be uniform, non-uniform or adaptive. Using the previously quantized pel value as a prediction, companded 5-bit quantization achieves imperceptible distortions in monochrome television.

The predictor enables the variance and correlation of the differential signal to be significantly less than that of the original signal, enabling the quantization noise to be reduced for a given number of quantization levels. Further gains can be made by *entropy coding* the quantized differential signal, i.e., assigning short code words to small, but frequently occurring, values and longer code words to the seldomly occurring large values. However, with entropy coding, the bit-rate depends on the input signal, and unless protective measures are taken there is a chance of some signals producing a bit-rate that exceeds the channel capacity causing severe distortion in the recovered signal.

In some cases it pays to represent *groups* of pels with a single code word. For example, with black/white graphics and text, long strings of identical pels occur both with and without predictive encoding. Considerable savings occur from coding such strings with a single binary word, called *run-length-coding*. Entropy coding of runs yields further gains. In television, conditional frame replenishment coder-decoders use previous frame prediction in non-moving areas of the picture. These areas are efficiently encoded as groups of pels.

DPCM performance can be improved significantly by using adaptive predictors, adaptive quantizers or both. Adaptive predictors attempt to optimize the prediction depending on the local waveform properties. For example, interframe coders typically use previous frame prediction in non-moving areas,

but in moving areas use linear combinations of pels in both the previous and present frame as a prediction. By adapting the moving-area predictor to the speed and direction of motion, further improvement is achieved. Adaptive DPCM (ADPCM) systems are ultimately limited by the predictor making predictions from pels corrupted by quantization noise. Thus, the design of the predictor should take into account the characteristics of the quantizer and vice versa.

With adaptive quantization, the effective number of quantization levels for a given bit-rate and quality of encoding is greatly increased. Adaptive quantizers used in picture encoding usually span the range of the input signal, and adaptively discard some of their levels as a function of the video signal. Sharp transitions in the waveforms need not be represented as accurately as when variations are relatively slow, i.e., visibility of quantization noise is markedly less at edges of objects than in flat, low detail areas. Such subjective phenomena enable adaptive quantization to save a bit or more per Nyquist sample. With a two-dimensional predictor, adaptive quantization and entropy coding, a good quality picture can be obtained with 1.5–2 bits/Nyquist sample. For conference scenes containing low motion, interframe motion compensated prediction is used with adaptive quantization and entropy coding. Only 0.1–0.2 bits/Nyquist sample are sufficient for good quality picture.

Correlations and periodicities can also be exploited by transform coding. With this approach the pels to be coded are first partitioned into blocks. Pels within a block need not be contiguous in time, i.e., groups of pels may be chosen from adjacent lines and adjacent frames in order to make up a block. Each block is then linearly transformed into another domain, e.g., frequency, having the desirable property that signal energy is concentrated in relatively few transform coefficients compared with the number of pels in the original block. Furthermore, all of the coefficients need not be quantized to the same accuracy to achieve a given quality of reproduction. By encoding only the significant coefficients with an accuracy dependent on human perception considerable bit-rate reductions are possible.

With adaptive transform coding (ATC), the coefficients selected for transmission change from block to block as does the quantization strategy for each coefficient. For single pictures, many blocks contain little or no picture detail, i.e., they contain energy only at low frequencies and require only a few bits for encoding. Other blocks contain high frequency components and produce more coding bits. Pictures containing an average amount of detail can be encoded using intraframe ATC at around 2 bits per pel with imperceptible distortion, i.e., *excellent* quality, and 1.5 bits per Nyquist sample with perceptible but not annoying distortion, i.e., *good* quality. In television three dimensional ATC operates on blocks from several frames yielding large reductions in bit-rate, particularly in non-moving areas of the picture.

Hybrid encoding involves transform coding of blocks of samples followed by block-to-block DPCM encoding of the resulting coefficients. Used in picture encoding, its performance is similar to transform coding with larger blocks; however, implementation is simpler. Interframe adaptive hybrid coding of pictures with low movement has yielded bit rates of 1 bit per Nyquist sample with *excellent* quality and 0.5 bits per sample with *good* quality[11.4].

ATC, ADPCM using entropy coding, and interframe coders all have the property that for a fixed quality, the bit-rate depends very much on the input waveform. This is undesirable for communication channels with a fixed channel capacity. Buffers can be used to accommodate the variable bit-rate generation to the constant bit-rate transmission. However, they introduce delay that may be intolerable in certain two-way communications. This may be reduced by sacrificing quality during periods of excessive bit-rate generation. Interframe coders take this approach by reducing moving-area picture quality during periods of rapid movement. In ATC it is customary to adaptively assign the bits to the coefficients in order to ensure constant bit-rate, i.e., allow the quality to vary from block to block. It must be emphasized that all constant bit-rate waveform codecs produce a variable quality either on a block basis, as with ATC, or on a per pel basis, as with DPCM. Another solution is for many encoders to share the same data channel, thus taking advantage of the low likelihood of several encoders producing high bit-rates at the same time.

Finally we note that subjective testing of DPCM speech and picture signals reveal some intriguing comparisons[11.5,11.6]. The effect of slope overload is to cause muffling in speech and blurring of the edges in pictures. With no slope overload, the quantization noise is often referred to as granular noise because it is perceived visually to have a granular texture. For a given mean noise power, slope overload noise is generally less annoying than granular noise for both pictures and speech. Also, moderate signal echo, i.e. reverberation in speech and ghosts in pictures, is usually more acceptable for a given distortion power than quantization noise. Interestingly, both listeners and viewers tend to have similar high order processing behavior. They perform a judgement as to whether the speech or picture is clear or unclear, and in the latter case whether the distortion is associated with the signal (slope overload, echo, bandlimiting) or the background (*snow* in pictures, hiss in audio).

11.4 Model-based Coding

In model-based coding of speech (known as vocoding), the signal is analyzed in terms of a model of the vocal mechanism, and the parameters of the model transmitted. The receiver then uses the parameters to synthesize a speech signal that is perceptually similar to the original speech. Unlike waveform encoding, no attempt is made to preserve the integrity of the individual samples of the waveform. There is no equivalent of vocoding in picture encoding because of the difference in the nature of the sources. In several constrained

situations, it is possible to construct models of the scene. For example, in videophone applications, most scenes contain a head-and-shoulders view of a person covering a static background. Head-and-shoulders views of people have been modeled as wire-frames with some success. This type of work is still in its infancy and faces formidable challenges in building models, finding applications, and dealing with the complexity of the encoder/decoder. Nonetheless, promising research is continuing.

11.4.1 Model-based Fax Coding

For black/white text consisting of typed characters, optical character recognition (OCR) algorithms identify each character and its position as two parameters that are then encoded and transmitted. This gives large compression gains compared to waveform encoding. Typically a standard 8.5×11 inch typewritten page (80×66 characters) can be transmitted with 8-bits per character, i.e., about 0.01 bits/Nyquist sample*. Black/white graphics can also be handled using OCR, but the character alphabet must be constructed adaptively for each document or class of documents to be coded. This usually necessitates the transmission of side information. Test documents have been encoded at ≈ 0.025 bits/Nyquist sample[11.4].

11.4.2 Model-based Coding of Video

As mentioned earlier, advances in computer vision and graphics are allowing us to progress from pel oriented coding (or waveform coding) to coding of objects in the scene. Techniques of computer vision can be used to identify a variety of objects from a complex scene. Each of these objects may contain hundreds of pels and can be described by a modeling process available from computer graphics.

In order to achieve a high degree of compression, proper models must be used. Currently, models that can handle arbitrary scenes do not exist. Therefore, two approaches are being pursued. In the first approach, a very limited class of scenes is considered in which the dominant object (e.g. human face) can be modeled easily. In the second approach, instead of constructing specific models, an image is segmented into regions using techniques from computer vision, and each region is coded separately. Regions are not necessarily rectangular blocks of pels as in block-transform coding. A brief description of each of these approaches is given below.

---

* We assume a sampling rate of 60 per mm^2. This is one of several rates in use and corresponds to *high resolution*. Also, "Nyquist sample" is not strictly correct since bandlimiting is not usually done.

One of the objects that is dominant in videophone applications is a human face. Modeling of human faces has been studied for some time[11.7-11.9]. Recent methods require only a few parameters in the model and yet create good quality images in most situations, resulting in good compression. Fig. 11.2a shows a block diagram of a model-based encoder. Each frame of the incoming video is analyzed with the assumption that the principal object in it is a human face. Computer vision techniques are used to extract and parametrize properties such as the shape of the head, geometric properties, color shading and texture. These parameters are used by both the transmitter and the receiver to create a three-dimensional model of the human head. As the head moves from frame to frame, these parameters are updated. The 3-D motion of the head is also estimated by techniques similar to the ones defined in Section 5.2. Based on the motion and the model parameters, both the transmitter and receiver reconstruct each frame. At the transmitter this frame is subtracted from the input frame, and the residue is coded by standard waveform coding techniques. At the receiver, the reconstructed image is added to the decoded residue image to create the image to be displayed.

The principal difficulty in this approach is to construct a three-dimensional model of the human face using the least number of parameters. Further, the model should be easily manipulated and easily synthesized by the receiver into an image that is a close replica of the input image. Wireframe models with triangular patches[11.8,11.9] are widely used for these reasons. See Fig. 11.2b.

Having found the model parameters, facial expressions need to be characterized. The Facial Action Coding System (FACS) developed by Ekman and Friesen[11.7] has a set of about 60 parameters that describe facial expressions. Wireframe models and FACS have been integrated by a group at Linkoping University led by R. Forcheimer[11.10]. The transmitter must estimate the specific values of each of these facial expression parameters for each frame and include them in the model parameters that are transmitted. The receiver then uses them to synthesize the image. Thus, the overall head motion is estimated by the 3-D motion estimation techniques similar to those of Section 5.2, whereas animation of the face is handled by FACS. As long as the unknown objects, background, etc., that are coded by the residue coder require a small fraction of the coded bits, the model based coder shows improvements in performance.

The second approach does not require any a priori knowledge or assumption about specific objects in the scene, and therefore can be applied to general classes of scenes. While there are many variations in this fast evolving area[11.11], the general characteristics are similar, and the differences are in the details. Basically, each frame of video is described in terms of moving objects and coded by three sets of parameters consisting of color (shading), contour (boundaries of objects) and motion.

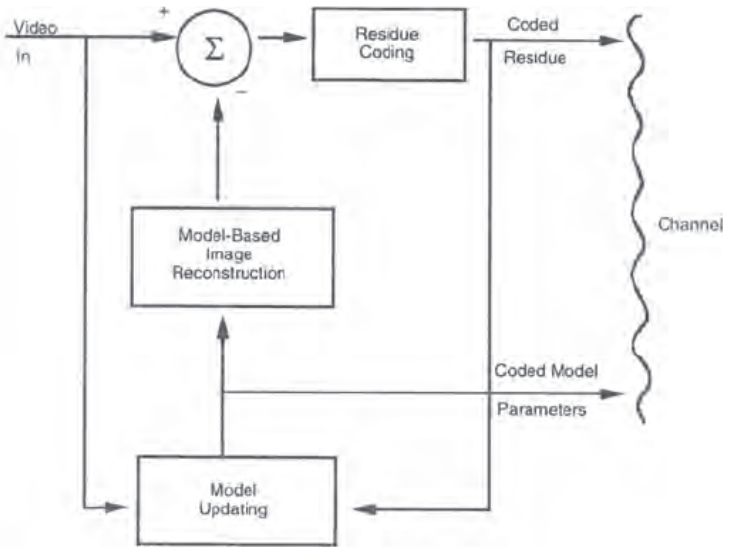


Fig. 11.2a Model Based Encoder.

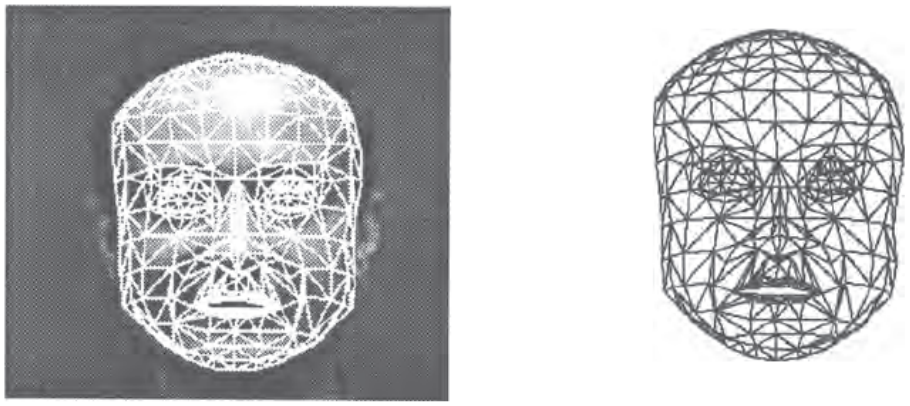


Fig. 11.2b Wire Frame Model.

A region-based object coder is shown in Fig. 11.3. Techniques from computer vision are used to segment each new frame into regions containing similar color or texture pels. Spatial and temporal consistency as well as edge continuity are used to generate well-defined contours. Both the new uncoded frame and the decoded frame (also available at the receiver) are necessary to maintain temporal consistency. Each region is then coded by standard waveform coding methods (e.g. DCT of motion compensated prediction error) specialized for non-rectangular regions.

The motion estimation, transform, quantization and rate-control all must be adapted for pel-regions of arbitrary shape. Many schemes have been proposed for this. In addition, unlike in the case of rectangular blocks, region boundaries must be coded. Here again many variations exist (for example, see [11.12], [11.13]).

The principal difficulty in region-based schemes described above is the ability of the algorithm to segment the image into areas of importance consistent with the regions perceived by the human eye. Additionally, for efficiency of compression, fewer regions are desirable. Extensive research is being carried out currently with the hope that impressive gains in performance can be made.

11.5 Bit Rates Versus Complexity

Fig. 11.4 is our attempt to summarize where we are today in picture coding. To accommodate sources of different bandwidths we use bits per Nyquist sample (η), and plot this as a function of codec complexity for different quality criteria. To convert η to bit-rate the following multiplication factors (sampling rates) apply: 288 line videotelephone –3 MHz, 525 line TV – 8.4 MHz, 625 line TV – 10 to 12 MHz. TV resolution facsimile is about 512×512 samples per picture; photographic resolution and black/white graphics use about 60 samples per mm^2.

The complexity allocated to each codec should not be taken too literally; rather it is a crude estimate relative to the complexity of an adaptive delta modulation (ADM) codec. The relation of cost to complexity is governed by an evolving technology, and codecs with high complexity that are prohibitively costly today may be inexpensive in the future. The different qualities of reproduction are more meaningful than the level of complexity. However, qualities are subjective measures and open to dispute, particularly for the newer codecs where rigorous subjective testing and comparisons are still to be made. The top curve corresponds to broadcast television quality, the middle curve to cassette videotape quality and the bottom curve to a quality that is usable, but with visible distortions. The dashed line corresponds to black/white text and graphics.

For single monochrome pictures (gray scale facsimile) the bit-rates can be reduced by roughly 20% to account for the fact that the "frozen noise" that appears in a coded single picture is much less visible than the time varying noise

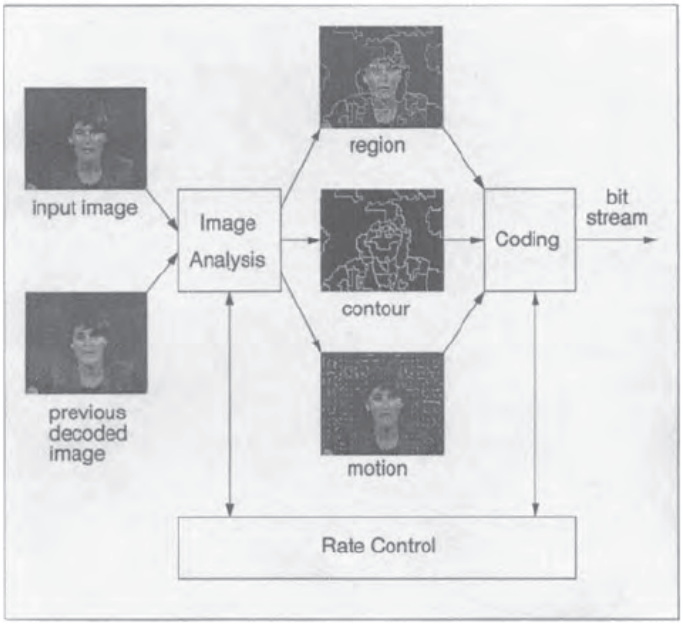


Fig. 11.3 Region Based Encoder.

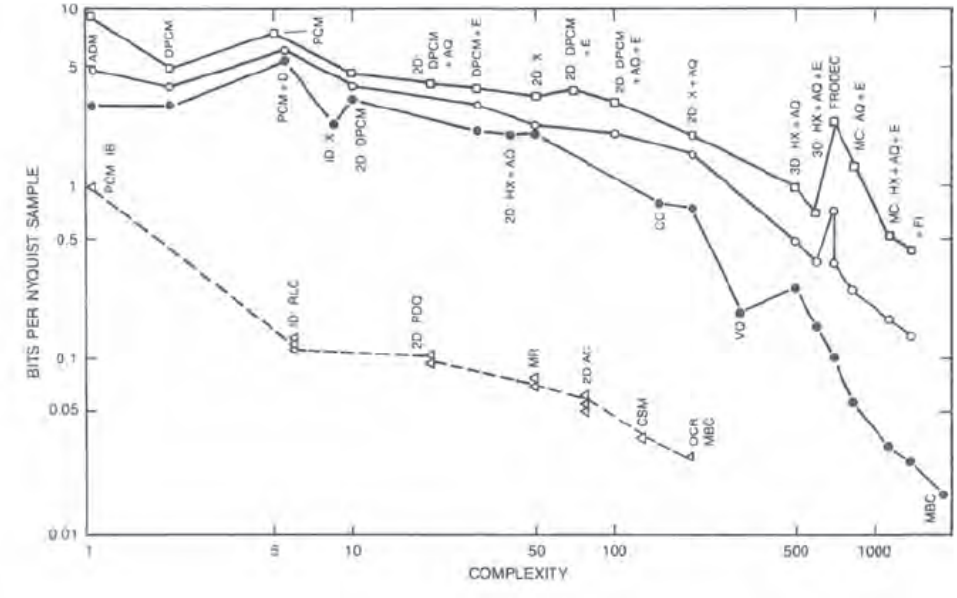


Fig. 11.4a Video bits per Nyquist sample η vs Complexity for monochrome television and black/white graphics. The three picture qualities correspond roughly to peak-signal-to-rms-unweighted-noise ratios (PSNR) of 45 dB (excellent), 40 dB (good), 35 dB (fair).

ADM	Adaptive Delta Modulation
DPCM	Differential pulse code modulation (previous element)
PCM	Pulse code modulation
PCM+D	PCM plus dither
1D:X	One dimensional transform coding
2D:DPCM	Two dimensional DPCM
2D:DPCM+AQ	2D:DPCM plus adaptive quantizing
DPCM+E	DPCM plus entropy coding
2D:HX+AQ	Two dimensional hybrid transform coding plus adaptive quantization.
2D:X	Two dimensional transform coding
2D:DPCM+E	2D:DPCM plus entropy coding
2D:DPCM+AQ+E	... plus adaptive quantizing and entropy coding.
CC	Contour coding (two component model [11.6])
2D:X+AQ	2D:X plus adaptive quantizing (JPEG Standard)
VQ	Vector Quantization
MC:AQ+E	Motion compensation, adaptive quantization and entropy coding
3D:HX+AQ	Interframe hybrid transform coding plus adaptive quantizing.
3D:HX+AQ+E	... plus adaptive quantizing and entropy coding
MC: HX+AQ+E	... with motion compensation (P*64 Standard)
+FI	... with spatial and temporal subsampling and interpolation
FRODEC	Conditional Frame replenishment coder-decoder
MBC	Model Based Coding
PCM:1B	PCM with one-bit (two level) quantizing
1D:RLC	One dimensional run length coding. (1D - CCITT standard)
2D:PDQ	Two dimensional predictive differential quantizing
MR	Modified READ code (2D - CCITT standard)
2D:AC	Two-dimensional Arithmetic Coding (JBIG Standard)
CSM	Combined Symbol Matching
OCR	Optical character recognition
MBC	Model-Based Coding

Fig. 11.4b Codec abbreviations in Fig. 11.4a.

or distortion in coded television. For color pictures the bit-rates should be increased by 15 to 25%.

Points toward the left in Fig. 11.4 are generally reliable. The algorithms have been studied in great detail, with a large variety of pictures and, in most cases, subjected to rigorous subjective testing. This tends to be less so for the more recently studied codecs of higher complexity, and many workers in the field (including the authors) would disagree with some of these published results. The non-monotonic nature of the curves in this region is probably attributed to the lack of refinement of the most complex systems and to the lack of detailed subjective evaluation.

11.6 Future Directions and Ultimate Limits

The usual approach to predicting the future is to extrapolate from the immediate past. The pitfalls of this are many. Constraints that we understand poorly or are completely unaware of will come into play eventually in an unpredictable manner. Unforeseen developments or inventions may occur that invalidate many, if not all, of the old assumptions. Also, the temptation to extrapolate techniques that, even today, are not well understood is usually too much to resist. In spite of all these pitfalls, we present the following predictions.

For some of the codecs discussed here we could have offered as performance limits information theory results[11.14–11.16] of SNR versus bit rate for input signals having stationary statistical properties. Unfortunately, and in spite of its wide use, SNR (with its numerous definitions) is a coarse yard-stick. A system with a high SNR will in general give good performance, but different codecs with their individual signal impairments, having similarly high SNR's will often be perceived quite differently. Low SNR (< 10 dB) values are meaningless in terms of codec performance. Fig. 11.4, therefore, relates to perceptual observations, not objective measures, and it is in perceptual terms that we must think of limits in the transmission bit-rate.

11.6.1 Waveform Encoding

In picture waveform encoding[11.3], intraframe ADPCM will be refined to the point where edges of objects are tracked by the prediction algorithm and quantizers adapt to the local picture characteristics in order to optimize the quantization noise according to subjective criteria. However, in a quasi-optimal system using entropy coding, edges contribute to overall bit-rate in varying amounts depending on the picture. With intraframe transform encoding the same may be said. Since most systems must accommodate both high and low detail pictures, bit-rates should not fall drastically.

Interframe ADPCM has advanced to the point where object motion is tracked by means of prediction. Adaptive filters and quantizers will optimize the displayed resolution (temporal and spatial) and quantization noise to the subjective requirements of the viewer. With these techniques, camera motion

(zooming and panning) will have little effect on overall bit-rate, with the result that the middle and upper curves of Fig. 11.4 will be essentially parallel to the lower curve.

In an interframe encoder with entropy coding, the long-term average and the short-term peak bit-rates differ considerably. For purposes of digital recording this is of little consequence. However, for present day real-time communication, where data peaks cannot be "buffered out" via the use of large memories and long delays, either excess channel capacity has to be provided or picture quality has to be compromised. In the future, there will be considerably more video traffic making it feasible for many video sources to share the same communication channel. Advantage can then be taken of the fact that simultaneous data peaks in several sources rarely occur, and the allocated *per source* channel capacity can be made much closer to the long term average data rate without introducing long delays due to buffering.

Ultimately, such channel sharing arrangements (equivalent to packet switching with variable length packets) appear to be the only way that real-time video can take advantage of highly adaptive waveform coders that produce low bit-rates for low detail or low movement pictures, but require higher rates otherwise. While it may be true that the "average" picture has average detail and average movement, few systems will be successful unless they can accommodate the full range of pictures that the average *viewer* finds interesting.

11.6.2 Model-Based Coding

Model-based coding has the prospect of further reductions in bit-rate compared to waveform coding. We consider model-based coding to include recognition of one or more global attributes of the objects in the image that enable more efficient coding while still achieving the quality objective.

For black/white text, the ultimate in parameter coding is character recognition. The visual information consists of symbols from a finite alphabet (which may be large as in the case of Chinese characters) and once identified they are easily coded in an efficient manner using entropy coding. For example, English text has an entropy between two and four bits/symbol.[11.17] Thus, an 8.5×11 inch typewritten sheet, double spaced, with one inch margins, containing about 1660 symbols requires about 5000 bits for coding or $\eta = 0.0014$ bits per pel at normal sampling rates. Multiple fonts and character sizes do not increase the bit-rate significantly.

In the future, the recognition aspect of text encoding will disappear. Instead, coded characters will pass directly from source terminal to local storage module or centralized data store, and will be displayed in soft or hard copy form only as needed.

For black/white non-text graphics, pattern recognition is also the key to ultra-efficient coding. However, this approach rapidly merges with the science of computer generated graphics and the study of specialized graphics languages.

Generally, the more specialized the language, the greater the efficiency of representation. For example, the mathematical expression

$$\left[\mathbf{A}(i,j,k)\right]_{a=0}^{a=5} = \int \int_{x+3}^{\frac{y}{2}} \left\{ \frac{g(x,y)+cos(2\pi ft+\phi)}{\tan^{-1}[h(v,t)]} \right\} dt\ d\theta$$

requires 1386 bits for specification under NROFF, the UNIX® word processing language. This is about $\eta = 0.0037$ bits/sample at 60 samples per mm^2.

Many other specialized graphics languages currently exist. For example, integrated circuit layouts never reside in computer memory in pel-by-pel form. Instead, they are built up from basic blocks according to instructions written in graphics language. Implementation of a generalized graphics language that will handle any and all forms of input is a long way away. However, we believe it is only through this approach that the ultimate limits of graphics encoding will be achieved.

The graphics language approach will ultimately benefit the encoding of gray scale pictures as well. Complete modeling of video scenes will probably not be possible, except for specialized situations like cartoons, human faces, etc. However, important features such as boundaries, locations and motion of objects are essentially graphical information, and is possible to represent them as such with very efficient codes. Motion of objects as well as shape changes can be representable by fairly efficient parameter codes.

Replication of the detail, shading, etc., within boundaries of objects requires additional *residue* data to be transmitted. The *amount* of residue needed depends on the application. In some cases, e.g. surveillance, very little is necessary; however, in broadcast television full cosmetic restoration must be maintained. The *fidelity* of the residue depends on context. For example, the texture in a grassy field may well be replaceable with random patterns having similar statistics, whereas detail in a human face may have to be replicated exactly.

Trying to estimate the ultimate bit-rate achievable by model-based coding of pictures suffers the same problem as with waveform coding, namely different pictures produce different bit-rates, and different applications require different fidelities of reproduction. In addition, how closely and what fraction of the image can be approximated by good models becomes additional factors. However, from the dashed curve of Fig. 11.4, the graphical parameters should require on average about 0.08 bits per pel for a single picture and perhaps a quarter of that (or less) for moving video depending on the amount of motion rendition required. The bit-rate needed for residue is much more elusive due to the large variation in pictures and fidelity requirements. We guess it should range downward from 1 bit per pel for excellent quality single pictures and a

quarter of that (or less) for moving video where interframe redundancy can be exploited.

11.6.3 Ultimate Display

The ultimate display would be a projection onto a large enclosed spherical surface with the viewer positioned at the center. The picture would fill the entire field of vision and, with appropriate control of temperature, odors, audio, gravity, etc., viewers would be unable to tell that they were not witnessing the original scene. Assuming human spatial vision cutoff at 60 cycles per degree, Nyquist sampling at 120 per degree yields about 594 million samples for the entire sphere. Further, assuming a picture presentation rate of 60 Hz, PCM luminance quantization of 10-bits, 30% additional bit-rate for chrominance and 75% additional bit-rate for stereo 3-D, the ultimate display represents a bit-rate of about 810 Gigabits per second.

11.7 Concluding Remarks

To extrapolate the future of picture compression encoding we have reviewed the current situation by classifying the performance of codecs in terms of bit-rate per Nyquist sample, complexity and perceptual quality. Both audio and video stimuli are electrical signals, and many encoding techniques, particularly those involving waveform encoding, are similar for both types of signals. This may seem somewhat surprising considering the radically different nature of the sources and the perceptions of audio and visual phenomena. The more efficient the compression algorithm, the more it is likely to be specific and unique to either speech or pictures. The ease at which encoding engineers can currently move between encoding audio and video signals, making comparatively minor concessions to source statistics and perceptual observations, demonstrates that substantial improvements are to be made. Picture compression is the interface between us and electronic systems, and the limits of encoding performance are ultimately set by how well we understand human perception, that is, ourselves.

References

[11.1] J. S. Goodman and D. E. Pearson, "Multidimensional Scaling of Multiply-Impaired Television Pictures", IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-9, No. 6, June 1979, pp. 353-356.

[11.2] N. S. Jayant, Ed., "Waveform Quantization and Coding", New York: IEEE Press Selected Reprint Series, 1976.

[11.3] A. N. Netravali and J. O. Limb, "Picture Coding: A Review", Proc. IEEE, Vol. 68, No. 3, March 1980, pp. 366-406.

[11.4] W. K. Pratt, Ed., "Image Transmission Techniques", Academic Press, New York, 1979.

[11.5] B. J. McDermott, C. Scagliola and D. J. Goodman, "Perceptual and Objective Evaluation of Speech Processed by Adaptive Differential PCM", Bell Syst. Tech. J., Vol. 57, 1978, pp. 1597-1618.

[11.6] P. Ekman and W. V. Friesen, "Facial Action Coding System", Consulting Psychologist Press, 1977.

[11.7] S. M. Park and N. I. Badler, "Animating Facial Expressions", IEEE Computer Graphics, Vol. 13, August 1981, pp. 245-252.

[11.8] F. I. Parke, "Parameterized Models For Facial Animation", IEEE Computer Graphics, Vol. 14, November 1992.

[11.9] R. Forcheimer and T. Kvonander, "Image Coding - Frame Waveforms to Animation", Vol. 37, No. 12, December 1989, pp. 2008-2023.

[11.10] H. G. Musmann, H. Hoffer, and J. Osterman, "Object-Oriented Analysis-Synthesis Coding of Moving Images", Signal Processing: Image Communication, Vol. 1, 1989, pp. 117-138.

[11.11] H. H. Chen, C. Horne, and B. G. Haskell, "A Region-Based Approach to Very Low Bit-rate Video Coding", (to be published), December 1993.

[11.12] M. Eden and M. Kocher, "On th Performance of a Contour Coding Algorithms in the Context of Image Coding - Part 1: Contour Segment Coding", Signal Processing, Vol. 8, 1985, pp. 381-386.

[11.13] J. K. Yan and D. J. Sakrison, "Encoding of Images Based on a Two-Component Source Model", IEEE Transactions on Communications, Vol. COM-25, No. 11, November 1977, pp. 1315-1322.

[11.14] H. G. Musmann and J. Klie, "TV Transmission Using a 64 kbit/s Transmission Rate", International Conference on Communications, 1979, pp. 23.3.1-23.3.5.

[11.15] A. D. Wyner, "Information Theory", Proc. IEEE, February 1981.

[11.16] C. E. Shannon, "Prediction and Entropy Printed English", BSTJ, Vol. 30, 1, January 1951; pp. 50-64.

Index