ADAPTIVE LOSSLESS VIDEO COMPRESSION

A Thesis

Submitted to the Faculty

of

Purdue University

by

Sahng-Gyu Park

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2003

In memory of my parents, Noh-Joon Park and Soo-Hyang Lee; To my parents-in-law, Byung-Kyu Cho and Yeon-Soo Doh.

ACKNOWLEDGMENTS

I would like to thank Professor Edward J. Delp for his academic guidance and professional advice. I would also like to acknowledge my Doctoral Committee: Professor Jan P. Allebach, Professor Michael D. Zoltowski and Professor Zygmunt Pizlo for their helpful comments and support. I am also grateful to Haoping Yu, who is part of the Corporate Research Group at Thomson in Indianapolis. Haoping and I worked together on a summer internship in 2002 where he gave me many inspirational comments about my research direction. I would also like to thank colleagues in the VIPER (Video and Image Processing) Lab for sharing time in discussions, especially Jinwha Yang who gave many helpful comments concerning research techniques.

My family has fully supported me throughout my years at Purdue. Jong-Hun (Bori) Park, Min-Seo (Sori) Park and especially my wife Jin-Hee Cho are my most valuable assets and I am eternally grateful for their emotional steadfastness. Last, but not least, I would like to thank the support of my brother and sisters in Korea.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Park, Sahng-Gyu. Ph.D., Purdue University, December, 2003. ADAPTIVE LOSSLESS VIDEO COMPRESSION. Major Professor: Edward J. Delp.

In this thesis, a new lossless adaptive compression algorithm for color video sequences is described. There are three types of redundancy in color video sequences: spatial, spectral and temporal redundancy. Our approach is to develop a new backward-adaptive temporal prediction technique to reduce temporal redundancy in a video sequence. The new temporal prediction technique is similar to the concept of the use of a motion vector, but requires that no motion vectors to be sent to the receiver. Another approach is to exploit both spatial and temporal redundancies in the video sequence. If there is a great deal of motion in the video sequence, temporal prediction does not perform well with respect to compression efficiency. In this case, only spatial prediction is used. In other cases, temporal prediction may work better than spatial prediction. An adaptive selection method between spatial and temporal prediction improves the compression performance. An adaptive integer wavelet transform is also investigated. Using the new backward-adaptive temporal prediction and the adaptive selection method between spatial and temporal prediction we show that our scheme is better than the state-of-the-art lossless compression algorithms.

1. INTRODUCTION

We live in a multimedia world. In this world, the storage and transmission of image and video data are driving the development of new compression methods. In international standards such as JPEG [1], JPEG-2000 [2], H.263 [3] and MPEG-4 [4] compression efficiency is achieved by sacrificing the quality of the image and video content. In some applications, the preservation of the original data is more important than compression efficiency, this is the area of lossless compression. Applications include storage and transmission of medical and astronomical images. Digital cinema archiving is another application of lossless video compression [5] [6].

There are three types of redundancies in color video sequences. These are spatial, spectral and temporal redundancy. Spatial redundancy exists among neighboring pixels in a frame. Pixel values usually do not change abruptly in a frame except near edges and highly textured areas. Hence there is significant correlation among neighboring pixels, which can be used to predict pixel values in a frame from nearby pixels. The predicted pixel is subtracted from the current pixel to obtain an error or a residual. The resulting residual frame has significantly lower entropy than that of the original frame. Lossless JPEG [1] uses simple linear predictors obtained by linear combination of neighboring pixels. CALIC [7] uses prediction by a nonlinear gradient. JPEG-LS [8] uses the MED (Median Edge Detector) prediction which can detect edges in neighboring pixels. These methods are described in detail in Section 3.2.

If a frame is in RGB color space, there are redundancies among the three color components. This is referred to as spectral redundancy. Penrose [9] uses a linear combination of neighboring pixels from a current color component and from a reference color component to predict the current pixel. Carotti [10] decorrelates color components using a differential coding scheme, where two of the colors are represented by the differences with the reference color. In the JPEG-2000 standard [2], a RCT (Reversible Color Transform) is used to convert from RGB to YCrCb color space. The RCT is a reversible integer-to-integer approximation of an irreversible RGB to YCrCb color space transform.

Consecutive frames in video sequence are very similar if no scene change has occurred. This is referred to as temporal redundancy. Temporal redundancy can be used to increase compression efficiency similar to the way spatial redundancy has been used in lossless image compression. Temporal redundancies between frames can be removed by temporal prediction. In lossy compression, block based temporal prediction by motion compensation provides very high coding gain where the technique decorrelates the frames. The data rate is further reduced by the quantization of the residual error frames. Motion vectors, needed to reconstruct the frame, are transmitted as overhead data. In lossless compression, the same technique can be used to predict pixel values. The performance using block based motion compensation is not as effective as in the case of lossy compression [11]. Several techniques utilizing temporal correlation in addition to spatial and spectral correlation have previously been described [9] [10] [11].

High motion areas have small correlation with corresponding areas in previous frames. In this case, temporal prediction does not work well and only spatial prediction can be used. Temporal prediction works better than spatial prediction in static areas where little motion has occurred between frames. It would seem logical that one may be able to increase the coding efficiency if one could use adaptive selection of the prediction technique. This concept will be used to develop our backward-adaptive temporal prediction technique described later.

The wavelet transform has been used as a successful tool for an image and video compression. One property of the wavelet transform that is important for compression applications is energy compaction. Most of the energy in the original image is compacted into the coarsest subband after the wavelet transform. The other finer subbands contain high frequency components of the original image and many wavelet coefficients in these subbands are small in magnitude [12]. Since the wavelet coefficients in the finer subbands are less correlated with each other, the prediction techniques utilizing three redundancies (spatial, spectral and temporal redundancies) does not perform satisfactorily. However, the wavelet coefficients in the coarsest subband are highly correlated each other and prediction techniques can remove the redundancies efficiently. Usually, the wavelet transform generates floating-point values, which means the operation is not reversible, although the original image can be recovered by the use of perfect reconstruction filters in principle. In lossless compression, the complete recovery of the original pixel values is required. Thus an IWT (Integer Wavelet Transform), which maps integer pixels to integer coefficients, can be used [13] [14] [15] [16]. The IWT can recover the original values of the image without any loss. It has been shown that an integer version of any floating-point wavelet transform can be obtained by the lifting scheme [17] [18]. This will be described in detail in Section 2.2.2. The performance of the IWT highly depends on the content of the image and video and the wavelet filter used.

The prediction error obtained by subtracting the prediction from the current pixel is encoded by adaptive arithmetic coding. In order to enhance the performance of the encoder, context modeling is used. An error is encoded using a different probability model determined by context. Probability models are updated as errors are encoded. Thus error can be more efficiently encoded by adaptive arithmetic encoding since the probability model represents the local characteristics of the image more accurately.

In Chapter 2, the basic concepts used throughout this thesis are presented. There is a brief overview of lossless and lossy image compression, image compression methods, integer wavelet transform, motion estimation and compensation and entropy coding. Chapter 3 describes various lossless image and video compression techniques in the literature including JPEG-LS and CALIC. A new lossless adaptive compression algorithm for color video sequences is described in Chapter 4. A new temporal prediction technique is developed. That is similar to the concept of the use of a motion vector, but requires no side information to be sent to the decoder. The adaptive selection of the prediction mode between temporal and spatial prediction based on the amount of motion in the video sequence is also investigated. A quantitative method to adaptively determine the proper IWT for a given frame in a video sequence is described. The decision is made by obtaining the uniformity and image activity [19] [20]. The performance comparisons with respect to the search range in temporal prediction is also described. A measure for video compressibility is defined. If the video compressibility is high for a video sequence, then the contribution of temporal prediction is high and the video sequence can be compressed well by the proposed algorithm. Finally, concluding remarks and future work are discussed in Chapter 5.

2. BACKGROUND

2.1 Lossless and Lossy Image Compression

The goal of image compression is to represent an image with the least amount of bits possible. There are “lossless” and “lossy” modes in image compression.

In lossy compression, the image quality may be degraded in order to meet a given target data rate for storage and transmission. The applications for lossy coding are the transmission of the image and video through a band limited network and efficient storage. The issue in lossy compression is how much we can reduce the degradation of the image quality given the data rate. In lossy compression, most algorithms transform pixels into the transform domain using the DCT (Discrete Cosine Transform) [21] or the DWT (Discrete Wavelet Transform) [22] [23] [24]. The source of the loss is either quantization of the transform coefficients or the termination of the encoding at a given data rate. In order to meet a data rate budget, the transform coefficients are explicitly quantized with a given step size as in ZTE [25], JPEG [1], H.263 [3] and MPEG-2 [26]. Implicit quantization is used in algorithms such as EZW [27] and SPIHT [28], which can be truncated at any point in the bitstream during encoding.

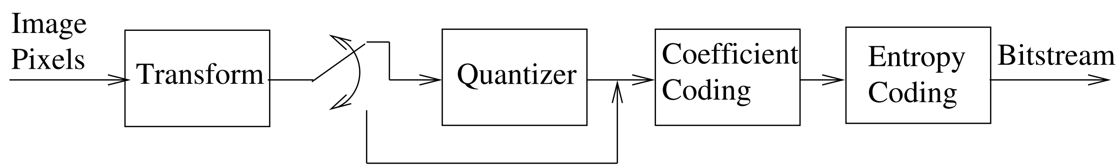
In the lossless compression, the image after decompression is identical to that of the original. The issue in lossless coding is how much we can reduce the data rate. The main approach for lossless image compression is predictive coding or entropy encoding. For predictive coding, DPCM (Differential Pulse Code Modulation) [29] is often used. Linear predictors are used where the prediction is obtained by the linear combination of previously decoded neighbors. For entropy coding, Run-length coding [30], Huffman coding [31], or arithmetic coding [32] [33] is used. Context-modeling can be included in the entropy encoding, which is to estimate the probability distribution function of the symbols conditioned on the context, so as to increase the compression performance. The context consists of a combination of neighboring pixels already encountered.

The structure of predictive coding is determined by the number of neighboring pixels used for the prediction, weighting for the linear combination of the neighboring pixels and the method of context modeling. The JPEG lossless mode [1] uses DPCM and Huffman coding or arithmetic coding. FELICS (Fast and Efficient Lossless Image Compression) [34] developed by P. Howard is a simple and fast lossless encoder. CALIC (Context-based, Adaptive, Lossless Image Coding) [7] by X. Wu and JPEG-LS [8] are other examples of lossless compression algorithms using predictive coding.

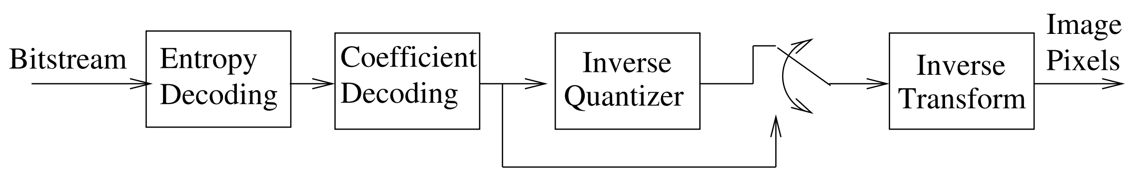
Another method for lossless encoding is to use the reversible integer-to-integer wavelet transform [35] [36]. The results of the integer wavelet transform are integers, so as to recover the originals completely with the inverse integer wavelet transform. The lifting scheme (LS) is used to implement the integer wavelet transform [14] [37].

2.2 Image Compression Methods

Many image compression algorithms use some form of transform coding. Figure 2.1 shows a block diagram of encoder and decoder using transform coding. The first step is to obtain a mathematical transformation to the image pixels in order to reduce the correlation between the pixels. The result of the transform is known as the *transform coefficients*. After this step, in lossy compression, an explicit quantizer may be used, or an implicit quantizer such as the truncation of the bitstream may be used. The source of the data loss in image compression is the quantizer. Thus, in the lossless compression case, the quantizer is not used. The third step is coefficient coding, which means that the transform coefficients are reorganized in order to exploit properties of the transform coefficients and obtain new symbols to be encoded at the fourth step. For example, the transform coefficients can be considered as a collection of quad-trees or zero-trees [27] [25] and/or treated in a bit plane fashion,



(a) Encoder



(b) Decoder

Fig. 2.1. Block Diagram of an Encoder and Decoder using Transform Coding.

so as to provide scalability to the compressed bitstream. The symbols from the coefficient coding are losslessly compressed at the entropy coding step. Entropy coding can be any method capable of compressing a sequence of symbols, such as Huffman coding [31], arithmetic coding [38] and Golomb coding [30].

2.2.1 Predictive Coding

For most images, there are redundancies among adjacent pixel values; that is, adjacent pixels are highly correlated. Thus, a current pixel can be predicted reasonably well based on a neighborhood of pixels. The prediction error, which is obtained by subtracting the prediction from the original pixel, has a smaller entropy than the original pixels. Hence, the prediction error can be encoded with fewer bits. In the predictive coding, the correlation between adjacent pixels is removed and the remaining values are encoded. Differential Pulse Code Modulation (DPCM) is a predictive coding technique. Let $x(n)$ be a pixel of an image and $\tilde{x}(n)$ be the prediction of $x(n)$. In the lossless DPCM case, $\tilde{x}(n)$ is predicted from previously decoded samples as in Figure 2.2:

$$\tilde{x}(n) = round\,[\sum_{i=1}^{N} \alpha(i)x(n-i)] \tag{2.1}$$

where N denotes the order of the linear predictor, *round* is a rounding function and $\alpha(i)$ for $i = 1, 2, \ldots, N$ are the prediction coefficients. The prediction error is:

$$e(n) = x(n) - \tilde{x}(n), \tag{2.2}$$

The prediction error is encoded by the symbol encoder. If the prediction is reasonably accurate, the prediction error $e(n)$ will be significantly smaller in magnitude than the magnitude of the original pixel $x(n)$. The dynamic range of an image is (0, 255) for 8 bits/pixel. The theoretic dynamic range of the prediction error is (-255, 255). Therefore the dynamic range of the signal to be encoded has been expanded. However, the histogram of the prediction error of a natural scene shows that it is highly peaked about 0 as in Figure 2.3 [39].

Figure 2.4 shows a lossy DPCM encoder and decoder. The prediction of $x(n)$, $\tilde{x}(n)$, is:

$$\tilde{x}(n) = \sum_{i=1}^{N} \alpha(i)\hat{x}(n-i) \tag{2.3}$$

where N is the order of the linear predictor. The prediction error is:

$$e(n) = x(n) - \tilde{x}(n), \tag{2.4}$$

If $\hat{e}(n)$ is the quantized version of $e(n)$ and $\tilde{e}(n)$ is the inverse quantized version of $\hat{e}(n)$, then the reproduced value of $x(n)$, $\hat{x}(n)$, is:

$$\hat{x}(n) = \tilde{e}(n) + \tilde{x}(n). \tag{2.5}$$

The decoder is simply the predictor loop of the encoder. The decoded pixel $\hat{x}(n)$ is:

$$\hat{x}(n) = \tilde{e}(n) + \tilde{x}(n). \tag{2.6}$$

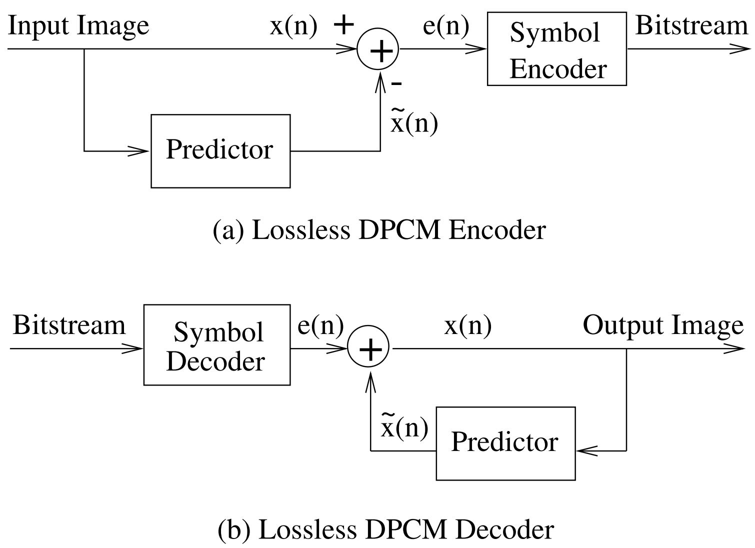


Fig. 2.2. Lossless DPCM Encoder and Decoder.

From Equation(2.4) and Equation(2.6), we obtain

$$\delta(n) = x(n) - \hat{x}(n) = e(n) - \tilde{e}(n). \tag{2.7}$$

Thus, the error $\delta(n)$ of the decoded pixel in the lossy DPCM case is exactly equal to the quantization error in $e(n)$.

2.2.2 Transform Coding

The goal of image compression is to represent the image pixels with the least number of bits possible. Thus, if the transform coefficients have a small number of non-zero values, they can be represented with a small number of bits. Therefore the transform needs to compact the energy into the fewest coefficients possible. Due to the property of the orthogonality of the transform, the average squared reconstruction error is equal to the average squared quantization error.

The KL(Karhunen-Loeve) transform produces the optimal energy compaction in the mean square sense by using the eigenvectors of the autocorrelation matrix [29]

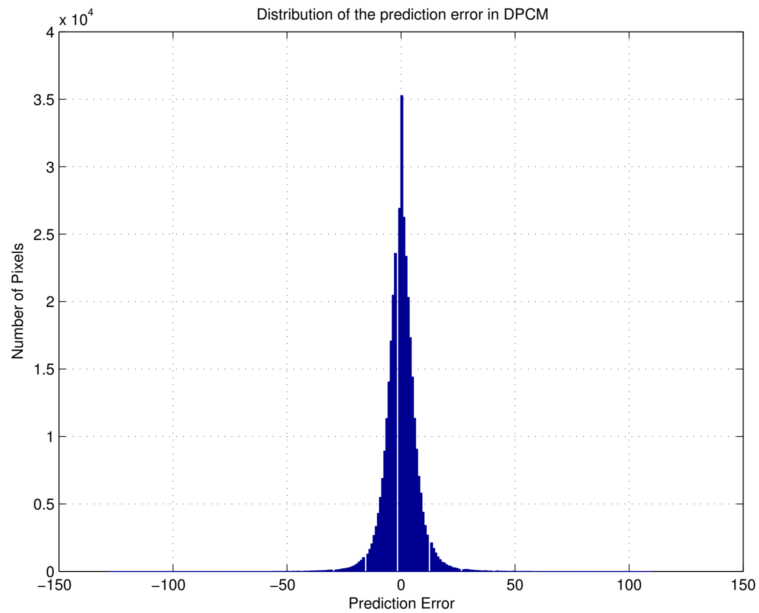


Fig. 2.3. Distribution of the Prediction Error in DPCM [39].

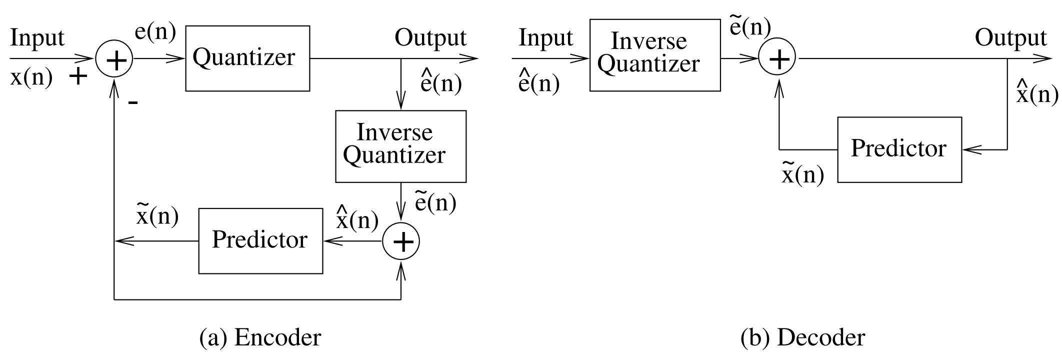


Fig. 2.4. Lossy DPCM Encoder and Decoder.

[40]. However, the use of the KL transform is difficult. First, the autocorrelation matrix of an image is not stationary. Second, the computation of the eigenvectors of the autocorrelation matrix may not be possible, since the autocorrelation matrix may be singular and some eigenvectors cannot be defined uniquely. Furthermore, the computation of the eigenvectors is very expensive and there is no fast KL transform algorithm. Therefore, the KL transform is usually used as a benchmark for other transforms, since it provides the lower bound of energy compaction.

Discrete Cosine Transform (DCT)

The DCT is related to the optimal KL transform in terms of energy compaction and there are fast DCT algorithms in 1 and 2 dimensions [21] [41]. These facts made the DCT the most popular transform for image compression. The forward 2-D DCT is defined as [42] [21] [43]

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos[\frac{\pi(2x+1)u}{2N}] \cos[\frac{\pi(2y+1)v}{2N}]. \tag{2.8}$$

For the inverse 2-D DCT is defined as

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} \alpha(u)\alpha(v) C(u,v) \cos[\frac{\pi(2x+1)u}{2N}] \cos[\frac{\pi(2y+1)v}{2N}], \tag{2.9}$$

where $\alpha(0) = \sqrt{\frac{1}{N}}$ and $\alpha(u) = \sqrt{\frac{2}{N}}$ for $u = 1, 2, \ldots, N-1$. The input image is $N \times N$ pixels. $f(x,y)$ is the intensity of the pixel in row x and column y. $C(u,v)$ is the DCT coefficient in row u and column v of the DCT matrix. The 2-D DCT is a separable transform, thus it can be computed by repeated application of 1-D DCT in horizontal and vertical directions in any order. Since an image is not stationary, the DCT is usually not used on the entire picture. The image is divided into $M \times M$ blocks and the transform is done on these blocks individually. Usually, the value of M is 8. This block processing method increases the compression performance. However, because of block processing, a *blocking artifact* may occur across the boundary of blocks in the reconstructed image. This is due to different errors between neighboring blocks and becomes visible especially at low data rates.

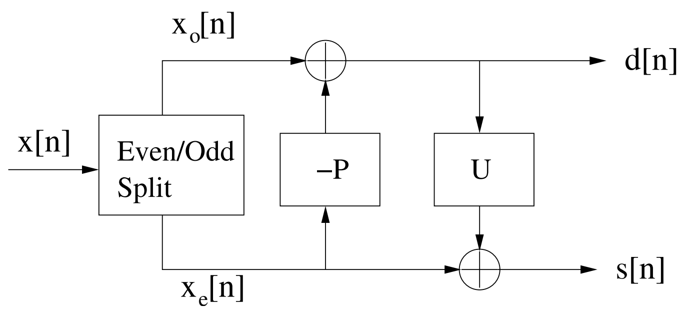


Fig. 2.5. Block Diagram of the Lifting Scheme: the Predict and Update Steps.

Integer Wavelet Transform

The wavelet transform generally produces floating-point coefficients. Although the original pixels can be reconstructed by perfect reconstruction filters without any loss in principle, the use of finite-precision arithmetic and quantization prevents perfect reconstruction. The reversible IWT (Integer Wavelet Transform), which maps integer pixels to integer coefficients and can reconstruct the original pixels without any loss, can be used for lossless compression [13] [14] [15] [16]. One approach used to construct the IWT is the use of the lifting scheme (LS) described by Calderbank et al [14]. The IWT construction using lifting is done in the spatial domain, contrary to the frequency domain implementation of a traditional wavelet transform [17] [18].

The basic idea behind the lifting scheme is to exploit the correlation among the signal samples. The correlation is typically local in space and frequency, i.e., adjacent samples and frequencies are more correlated than ones that are far apart [37]. A typical lifting scheme consists of the three steps shown in Figure 2.5: Split, Predict and Update [18].

Split: Let $x[n]$ be the input signal. The input is first split into even and odd indexed samples. The even indexed samples are $x_e[n] = x[2n]$ and the odd indexed samples are $x_o[n] = x[2n+1]$. Typically the two sets are closely correlated. Thus, a prediction of a value in one set can be made by using the other set.

Predict: The odd indexed samples $x_o[n]$ can be predicted by a predictor P from the neighboring even indexed samples $x_e[n]$. The predictor $P(x_e)$ is a linear combination of neighboring even indexed samples:

$$P(x_e)[n] = \sum_i p_i x_e[n+i]. \tag{2.10}$$

This is the "predict" step. This predictor need not be exact and the prediction error or detail $d[n]$ is:

$$d[n] = x_o[n] - P(x_e)[n]. \tag{2.11}$$

$d[n]$ has high frequency coefficients after the wavelet transform. We can expect the first order entropy of $d[n]$ to be smaller than that of $x_o[n]$, if the predictor P is accurate. The operation of obtaining a prediction and the detail is known as the "lifting scheme."

Update: For the even indexed samples $x_e[n]$, an "update" step by an update operator U is used to obtain smoothed values $s[n]$ with the use of the detail:

$$s[n] = x_e[n] + U(d)[n], \tag{2.12}$$

where $U(d)[n]$ is a linear combination of neighboring $d[n]$ values:

$$U(d)[n] = \sum_i u_i d[n+i]. \tag{2.13}$$

This is a second "lifting" step. $s[n]$ has low frequency coefficients after the wavelet transform.

The lifting scheme is invertible, thus no information is lost. The reconstruction algorithm follows the same structure as the lifting scheme, but in reverse order. i.e., $x_e[n]$ is first recovered by:

$$x_e[n] = s[n] - U(d)[n], \tag{2.14}$$

and then $x_o[n]$ can be obtained by:

$$x_o[n] = d[n] + P(x_e)[n]. \tag{2.15}$$

One example of the IWT is the S transform [14] [44], which is an integer version of the Haar transform. The forward S transform is:

$$\begin{aligned} d[n] &= x[2n+1] - x[2n] \\ s[n] &= x[2n] + \lfloor d[n]/2 \rfloor \end{aligned} \tag{2.16}$$

and the inverse S transform is:

$$\begin{aligned} x[2n] &= s[n] - \lfloor d[n]/2 \rfloor \\ x[2n+1] &= d[n] + x[2n]. \end{aligned} \tag{2.17}$$

The S+P transform proposed by Said and Pearlman [45] performs a S transform first, linearly predicts the lowpass coefficients $s_{1,l}$ and generates the highpass coefficients. The following is the forward S+P transform:

$$\begin{aligned} d_{1,l}^{(1)} &= s_{0,2l+1} - s_{0,2l} \\ s_{1,l} &= s_{0,2l} + \lfloor d_{1,l}^{(1)}/2 \rfloor \\ d_{1,l} &= d_{1,l}^{(1)} + \lfloor \tfrac{2}{8}(s_{1,l-1} - s_{1,l}) + \tfrac{3}{8}(s_{1,l} - s_{1,l+1}) + \tfrac{2}{8} d_{1,l+1}^{(1)} + \tfrac{1}{2} \rfloor. \end{aligned} \tag{2.18}$$

The inverse S+P transform is:

$$\begin{aligned} d_{1,l}^{(1)} &= d_{1,l} - \lfloor \tfrac{2}{8}(s_{1,l-1} - s_{1,l}) + \tfrac{3}{8}(s_{1,l} - s_{1,l+1}) + \tfrac{2}{8} d_{1,l+1}^{(1)} + \tfrac{1}{2} \rfloor \\ s_{0,2l} &= s1,l - \lfloor d_{1,l}^{(1)}/2 \rfloor \\ s_{0,2l+1} &= d_{1,l}^{(1)} + s_{0,2l}. \end{aligned} \tag{2.19}$$

Other IWT examples of the IWT are [14]:

- A (2,2) transform:

$$\begin{aligned} d[n] &= x[2n+1] - \lfloor \tfrac{1}{2}(x[2n] + x[2n+2]) + \tfrac{1}{2} \rfloor \\ s[n] &= x[2n] + \lfloor \tfrac{1}{4}(d[n-1] + d[n]) + \tfrac{1}{2} \rfloor. \end{aligned} \tag{2.20}$$

- A (2+2,2) transform:

$$\begin{aligned} d^{(1)}[n] &= x[2n+1] - \lfloor \tfrac{1}{2}(x[2n] + x[2n+2]) + \tfrac{1}{2} \rfloor \\ s[n] &= x[2n] + \lfloor \tfrac{1}{4}(d^{(1)}[n-1] + d^{(1)}[n]) + \tfrac{1}{2} \rfloor \\ d[n] &= d^{(1)}[n] - \lfloor \tfrac{1}{16}(-s[n-1] + s[n] + s[n+1] - s[n+2]) + \tfrac{1}{2} \rfloor. \end{aligned} \tag{2.21}$$

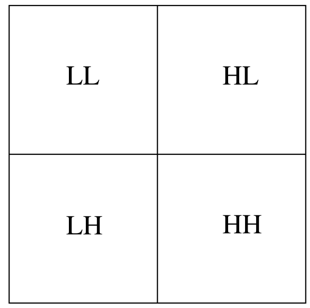


Fig. 2.6. Subbands after One Level of the IWT.

- A (4,4) transform:

$$\begin{aligned} d[n] &= x[2n+1] - \lfloor \tfrac{9}{16}(x[2n] + x[2n+2]) - \tfrac{1}{16}(x[2n-2] + x[2n+4]) + \tfrac{1}{2} \rfloor \\ s[n] &= x[2n] + \lfloor \tfrac{9}{32}(d[n-1] + d[n]) - \tfrac{1}{32}(d[n-2] + d[n+1]) + \tfrac{1}{2} \rfloor \end{aligned} \tag{2.22}$$

Here, the notation $(N, \tilde{N})$ represents a transform with N and $\tilde{N}$ vanishing moments in the analysis and synthesis high pass filters, respectively. These IWTs are separable, thus they can be used on images row-wise and column-wise in any order. One level of the IWT generates 4 subbands, LL, LH, HL and HH, as shown in Figure 2.6.

2.3 Motion Estimation and Compensation

Consecutive frames in a video sequence are very similar. This redundancy between successive frames is known as temporal redundancy. Video compression methods can exploit temporal redundancy by estimating the displacement of objects between successive frames (motion estimation). The resulting motion information (a motion vector) can then be used for an efficient inter-frame coding (motion compensation). The motion information along with the prediction error are then transmitted instead of the frame itself.

Block Matching (BM) techniques are the most common methods for motion estimation in video coding [46] [26]. Typically each block, known as macro block, (16×16 pixels) in the current frame is compared with displaced regions of the same size from the previous frame. The displacement which results in the minimum error is selected as the motion vector for the macro block. The prediction error frame is obtained by subtracting the motion compensated frame from the current frame. The best performance of the BM can be obtained by a full search that considers every possible movement of the macro block in the search area. However, the full search is very computationally intensive. For example, to analyze a search range of up to $\pm$ 15 pixels in the horizontal and vertical directions requires $31 \times 31 = 961$ shifts, each of which involves $16 \times 16 = 256$ pixel difference computations for a given macro block.

Many sub-optimal matching algorithms are proposed in the literature. Significant reduction of the computation can be achieved with hierarchical BM [47] [48], in which an approximate motion estimate is obtained from an exhaustive search using a lowpass subsampled pair of images, the estimate is then refined by a small local search using the full resolution images. Subsampling 2:1 in each direction reduces the number of macro block pixels and the number of shifts by 4:1, producing a computational savings of 16:1.

Typically many regions in a frame have different amount of motions. For regions with less motion a large block size is enough for motion estimation while a small block size is required for high motion regions. H.264 [49] [50] [51] utilizes different block sizes for motion estimation while many algorithms use fixed block size. The maximum block size is 16×16 and the minimum is 4×4. Rectangular blocks are also possible. Usually the block size cannot be very small since motion vectors need to be transmitted to the decoder as side information.

2.4 Entropy Coding

The last step in an image and video encoder is to encode the symbols losslessly using entropy coding. In order to represent an input symbol sequence efficiently, the symbols with a higher frequency of occurrence should be represented with fewer bits than symbols with a lower frequency of occurrence. This requires the need for a probability model of the input symbols.

2.4.1 Basic Information Theory

Information theory was introduced by C. E. Shannon [52] in 1948. Entropy is a measure of the information content of a source. Suppose we have a discrete random variable X with alphabet A_x and probability mass function $p(x) = Pr(X = x), x \in A_x$. The Shannon information of the outcome x is defined to be

$$h(x) = \log_2 \frac{1}{p(x)} = -\log_2 p(x). \tag{2.23}$$

The logarithm typically has base 2, measured in *bits*. The entropy of a discrete random variable X is defined to be the average Shannon information of an outcome and depends on the probability distribution of the outcome. The entropy $H(X)$ is [53] [54]:

$$H(X) = -\sum_{x \in A_x} p(x) \log_2 p(x), \tag{2.24}$$

with the convention that $0 \times \log_2 0 = 0$ for $p(x) = 0$. The entropy has its maximum at $H_{max} = log_2 M$ for $p(x) = \frac{1}{M}$ which means that all events have the same probability of occurrence and M is the size of the alphabet A_x.

The joint entropy $H(X, Y)$ of two discrete random variables (X, Y) with a joint probability distribution $p(x, y)$ is defined as

$$H(X, Y) = -\sum_{x \in A_x} \sum_{y \in A_y} p(x, y) \log_2 p(x, y), \tag{2.25}$$

which can also be viewed as the expectation of $\log_2 p(x, y)$.

The conditional entropy of a random variable given another can be defined as the expected value of the entropies of the conditional distributions, averaged over the conditioning random variable. The conditional entropy $H(X|Y)$ is defined as

$$\begin{aligned} H(X|Y) &= \textstyle\sum_{y\in A_y} p(y)H(X|Y=y) \\ &= -\textstyle\sum_{x\in A_x}\sum_{y\in A_y} p(x,y)\log_2 p(x|y). \end{aligned} \tag{2.26}$$

The mutual information $I(X;Y)$ is the relative entropy between the joint distribution $p(x,y)$ and the product distribution $p(x)p(y)$, i.e.,

$$I(X;Y) = \sum_{x\in A_x}\sum_{y\in A_y} p(x,y)\log_2 \frac{p(x,y)}{p(x)p(y)}. \tag{2.27}$$

The mutual information can be rewritten as

$$I(X;Y) = H(X) - H(X|Y), \tag{2.28}$$

which is the reduction in the uncertainty of X due to the knowledge of Y.

From the above results, the following can be derived:

$$\begin{aligned} &H(X,Y) \le H(X) + H(Y) \\ &H(X|Y) \le H(X). \end{aligned} \tag{2.29}$$

The joint entropy of X and Y is less than or equal to the sum of the individual entropies of X and Y with equality if and only if X and Y are independent. The conditional entropy of X given Y is less than or equal to the entropy of X with equality if and only if X and Y are independent. The conditioning reduces the entropy of a random variable on the average. This is the reason that context modeling is used in entropy coding. The context is a condition on the current symbol as a function of the neighboring symbols. The probability mass function of a symbol conditioned on a given context is estimated.

2.4.2 Huffman Coding

Entropy coding uses variable-length codewords that assigns a shorter codeword to a more probable symbol and a longer codeword to a less probable symbol. A practical method for constructing compact codes was provided by Huffman [31]. Huffman coding is widely used due to its speed and performance.

Assume that a source has an alphabet of size M. We construct a full binary tree, whose node has either zero or two children. Initially, there is a set of singleton trees with corresponding probabilities of the symbols. At each step in the Huffman coding algorithm, the trees corresponding to the two least probabilities are merged into a new tree whose root probability is the sum of the two children probabilities. The children nodes of the new tree are removed from further consideration. The remaining trees with the new tree are used for the next step and this is repeated until all trees are merged into one tree. There are many alternatives for specifying the actual codewords and it is necessary that the code has the prefix property [53]. The actual code assignment is performed by labelling the edge from each parent to its left child with the digit “0” and the edge to the right child with “1”. The codeword for each alphabet is the sequence of labels along the path from the root to the leaf node representing the alphabet.

Although Huffman coding is simple and fast, it has two drawbacks. The first is that the codeword lengths are integers. Thus the optimal performance can be obtained when all the symbol probabilities are integral powers of two, which is not common in images. It is especially inefficient for a binary source since each symbol must be coded with at least one bit, even though the entropy of the source is much less than one bit/symbol. The second is that Huffman coding does not adapt well when the statistics of the source changes. This can be addressed by using a two-pass algorithm where the first pass is to collect the symbol statistics and generate the Huffman tables and the second pass is to encode the symbols. However, there is overhead in transmitting the Huffman tables and the inability to adapt to short-term statistics.

2.4.3 Arithmetic Coding

The basic concept of arithmetic coding is that a given probability set, $[0, 1]$, assign an interval to every symbol based on its probability [32] [55] [38]. The higher the symbol probability, the longer the interval assigned to it. Assume that we have a sequence of symbols with finite alphabet and know the probability of each symbol. Suppose that p_i is the probability of the ith symbol and that L is the smallest probability interval, which the symbol sequence processed so far resides in and R is the probability interval length, which is the product of the probabilities of the symbols. L and R are initialized to 0 and 1 respectively. To encode the jth symbol from the sequence, both L and R must be updated: L is refined by $L + R\sum_{i=1}^{j-1} p_i$ and R is replaced by $R \cdot p_i$. After the entire sequence is processed, any value between L and $L + R$ will uniquely specify the input sequence. A number with the shortest binary representation in the interval will be transmitted. The algorithm is:

- $Low = 0$
- $High = 1$
- For all input symbols
  - $Range = High$ - Low
  - $High = Low + Range \times$ High range of the symbol being coded
  - $Low = Low + Range \times$ Low range of the symbol being coded,

where *Range* keeps track of where the next *Range* should be and *High* and *Low* specify the output probability intervals. Table 2.1 shows the symbol probabilities with 3 alphabets and Table 2.2 shows the arithmetic code for "$BACA$" symbol sequence.

The output number will be 0.59375. The decoding algorithm starts with the input number 0.59375 received. The first step is to find where the number 0.59375 corresponds to the probability interval $[0, 1]$ using the symbol probability table in

Table 2.1
Symbol Probabilities of 3 Input Alphabets.

Symbol	Probability	Range
A	1/2	[0, 0.5)
B	1/4	[0.5, 0.75)
C	1/4	[0.75, 1)

Table 2.2
An Example of Arithmetic Encoding.

Symbol	Range	Low value	High value
		0	1
B	1	0.5	0.75
A	0.25	0.5	0.625
C	0.125	0.59375	0.625
A	0.03125	0.59375	0.609375

Table 2.3
An Example of Arithmetic Decoding.

Number	Range	Symbol
0.59375	0.25	B
0.375	0.5	A
0.75	0.25	C
0	0.5	A

Table 2.1. The input 0.59375 is in $[0.5, 0.75]$ which corresponds to the symbol B. The symbol B is the first decoded symbol as shown in Table 2.3. Then the symbol range $[0.5, 0.75]$ for the symbol B is expanded into $[0, 1]$. In order to do this, the low boundary of the symbol range, 0.5, is subtracted from the input number, 0.59735 and the new number, 0.09375 is divided by the range of the symbol B to obtain the new input number 0.375. Now the decoding is repeated again with the input number 0.375 as in Table 2.3. The decoding algorithm is:

1. Output a corresponding symbol from Number

2. Using the Number and the symbol

   - $Range$ = High range of the symbol - Low range of the symbol
   - $Number$ = $Number$ - Low range of the symbol
   - $Number$ = $Number$ / $Range$
   - Output the corresponding symbol,

3. Repeat the step 2

where $Number$ is an input number transmitted from the encoder.

In arithmetic coding, a probability model is needed to accurately estimate the probability of the symbols in the input data. The better the model, the better the compression performance. By using finite context modeling, the compression efficiency can be improved. Finite context models assign a probability to a symbol based on the context in which the symbol appears.

For the above coding algorithm, the probability distribution of the symbols needs to be known and sent to the decoder as overhead. If the probability distribution changes, the encoder becomes sub-optimal and a new probability distribution needs to be obtained. Adaptive arithmetic coding is a solution to this problem. The probability distribution of the symbols is obtained by the encoder *on the fly* based on the symbols that have already been processed. The decoder can be synchronized to the encoder. With adaptive arithmetic coding, no probability distribution functions need to be transmitted to the decoder. For non-stationary data such as images, the encoder and the decoder can adapt their probability distribution to the changing data.

2.4.4 Golomb Coding

Golomb Coding [30] is a practical and powerful implementation of Run-Length Coding. Golomb codes are optimal for geometric distribution of nonnegative integers. This coding is characterized by an encoding parameter m that is constrained to be a power of two. The optimal value of m is the nearest power of two of $p \ln(2)/(1-p)$, where p is the probability of zero. The following is the coding procedure:

- Assume that there is a binary sequence and 0 is the more probable bit.

- Break the sequence into runs of zeros 0^i1, which means a run of i runs of ending with an one. The last bit is known as the *tail* bit.

- Code each 0^i1 as 1^q0y, where $i = qm + r$ $(0 \leq r < m)$ and y is the binary representation of r using $\log_2 m$ bits. Table 2.4 shows examples of Golomb codes. For example, a binary sequence, "00001", is represented as 0^41 where the length of zeros is 4, i.e., $i = 4$. If the encoding parameter, m, is 2, then $i = qm + r = 2q + r$ and y is represented by one bit. Since $i = 4$, $q = 2$,

Table 2.4

Examples of Golomb Codes. A Mid-point ($\cdot$) Separates the q and y Values in the 1^q0y Representation.

Run-Length of Zeros i	Encoding Parameter m		
	2	4	8
0	0·0	0·00	0·000
1	0·1	0·01	0·001
2	10·0	0·10	0·010
3	10·1	0·11	0·011
4	110·0	10·00	0·100
5	110·1	10·01	0·101
6	1110·0	10·10	0·110
7	1110·1	10·11	0·111
8	11110·0	110·00	10·000
9	11110·1	110·01	10·001
⋮	⋮	⋮	⋮

$r = 0$ and y="0". The Golomb code of the binary sequence "00001" is then 1^q0y = "1100". If m is 4, then $i = qm + r = 4q + r$ and y is represented by 2 bits. Since $i = 4$, $q = 1$, $r = 0$ and y="00". The Golomb code of the binary sequence "00001" is then 1^q0y = "1000". Similarly, the Golomb code of the binary sequence "00001" for $m = 8$ is "0100".

- The final bitstream is: MP_b $code_1$ $code_2$ $\ldots code_n$ *tail*, where MP_b is the 1 bit of the more probable bit value, $code_j$ is the code of the j-th run and *tail* is a single bit that is one if the last run has a tail and is zero otherwise.

3. REVIEW OF LOSSLESS VIDEO COMPRESSION

Lossless compression of images and video can be classified into two categories, transform based and prediction based techniques. In this section, lossless compression techniques are reviewed.

3.1 Lossless Image Compression Based on Transform Coding

Usually wavelet transform based lossless compression techniques generate embedded bitstreams, which can allow a lossy version of the image to be decoded for a given data rate. Hence the IWT (Integer Wavelet Transform) has both lossy and lossless compression modes. Performance evaluation of various IWTs constructed by lifting [14] is reported by Adams [56]. It is known that the performance of the discrete wavelet transform outperforms that of IWT at low data rates [13] [15]. In order to bridge the performance gap between lossy and lossless compression, "lossy-plus-residual" techniques have been described. The first coding layer is generated by the conventional DWT, a residue obtained by the difference between the original image and its lossy reconstruction is coded using integer wavelet packet transform for the second coding layer [57]. The transform coefficients may be encoded using EZW (Embedded Zerotree Wavelet) [58] or tree based SCP (Spatial Coefficient Partitioning) [59], so as to obtain an embedded bitstream.

3.1.1 JPEG-2000 Lossless Mode

JPEG-2000 [2] [60] is an ISO/ITU standard for still image coding. It provides functionalities for lossy and lossless compression of grayscale and color images. JPEG-2000 is based on EBCOT (Embedded Block Coding with Optimized

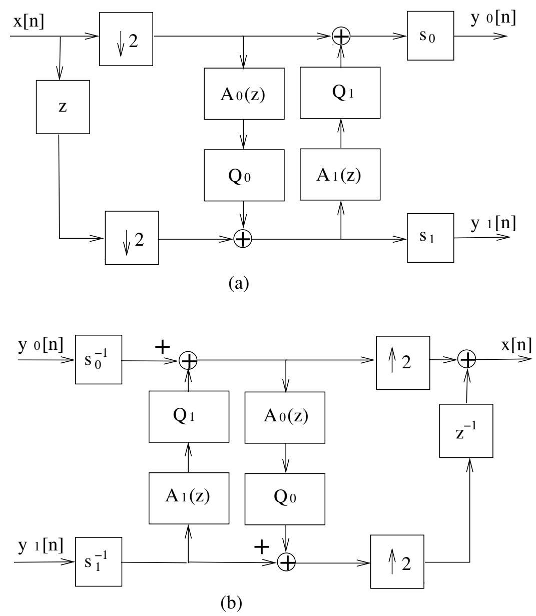


Fig. 3.1. Integer-to-Integer Wavelet Transform using Lifting in JPEG-2000. (a) Forward transform (b) Inverse transform.

Truncation) [61], which offers rate and spatial scalability with excellent compression performance. JPEG-2000 consists of the DWT, scalar quantization, bit plane coding, context modeling, arithmetic coding and post-processing for rate allocation. The DWT can either be a reversible integer-to-integer transform used for lossless coding, or be a non-reversible real-to-real transform, which offers higher compression with loss of information [62]. For the intra-component transform of the lossless mode, the reversible 5/3 integer-to-integer wavelet transform is used. This is an approximation of the linear wavelet transform [14]. Figure 3.1 shows the forward and backward wavelet transform using lifting with the following parameters:

$$
\begin{aligned}
&A_0(z) = -\tfrac{1}{2}(z+1),\ A_1(z) = \tfrac{1}{4}(1+z^{-1}), \\
&Q_0(x) = -\lfloor -x \rfloor,\ Q_1(x) = \lfloor x + \tfrac{1}{2} \rfloor,\ s_0 = s_1 = 1.
\end{aligned}
\tag{3.1}
$$

For color images, an intercomponent transform is used to reduce the correlation between color components before the reversible wavelet transform. The RCT

(Reversible Color Transform) is a reversible integer-to-integer approximation of the irreversible RGB to YC_rC_b transform widely used in video compression. The following is the forward RCT:

$$\begin{aligned} Y &= \lfloor (R + 2G + B)/4 \rfloor \\ C_r &= R - G \\ C_b &= B - G. \end{aligned} \tag{3.2}$$

The inverse RCT is:

$$\begin{aligned} G &= Y - \lfloor (C_r + C_b)/4 \rfloor \\ R &= C_r + G \\ B &= C_b + G. \end{aligned} \tag{3.3}$$

After the reversible wavelet transform, each subband is divided into rectangular blocks known as code-blocks, typically 32×32 or 64×64. The code-blocks are encoded independently of one another. Each block is entropy coded using context modeling and bit-plane arithmetic coding.

3.2 Lossless Image Compression Based on Spatial Prediction

There are two types of redundancies in a color image, spatial and spectral redundancy. The exploitation of spectral redundancy will be discussed in the next section. In prediction based lossless image compression, prediction methods are used to exploit the spatial correlation of the pixels. For most images, there are redundancies among adjacent pixels. Thus, a current pixel can be predicted reasonably well based on its neighboring pixels. The prediction error, which is obtained by subtracting the prediction from the original pixel, has a smaller entropy than that of the original pixel. Hence, the prediction error can be encoded with fewer bits. The JPEG lossless mode [63] [64] uses predictive schemes, which provides seven different predictors. The prediction uses up to 3 neighboring pixels as shown in Table 3.1 with A, B and C as specified in Figure 3.2. For example, if the prediction mode is 4, the prediction of $x(u, v)$ is $A + B - C$. The prediction error is encoded using context based arithmetic coding or Huffman coding.

Table 3.1
JPEG Predictors for Lossless Coding.

Mode	Predictor for x(u,v)
0	0 (No prediction)
1	B
2	A
3	C
4	A+B-C
5	A+(B-C)/2
6	B+(A-C)/2
7	(A+B)/2

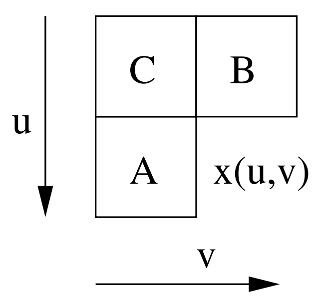


Fig. 3.2. Neighboring Pixels for the JPEG Predictors.

3.2.1 CALIC (Context-Based Adaptive Lossless Image Coding)

CALIC [7] [64] uses a large number of modeling contexts to condition a nonlinear predictor and adapt the predictor to varying source statistics. Figure 3.3 shows the block diagram of the CALIC encoder. It has two modes of operation: *binary* and *continuous-tone mode*. The *Binary mode* is used when neighborhoods of the current pixel to be encoded have no more than two distinct intensity values. In the binary mode, a ternary symbol including an escape symbol which causes a switchback to the continuous-tone mode is entropy coded by a context-based arithmetic encoder. In *continuous-tone mode*, the system performs four major steps: gradient-adjusted prediction (GAP), context selection and quantization, context modeling of prediction errors, and entropy coding of the prediction errors as shown in Figure 3.3. One of two operation modes is selected automatically during the coding process, depending on the context of the current sample.

GAP does prediction by a linear combination of neighboring pixels with respect to the estimated gradients in the neighborhood. The estimated gradient at the current pixel $x(u, v)$ is obtained by:

$$\begin{aligned} d_h &= |A - E| + |B - C| + |B - D| \\ d_v &= |A - C| + |B - F| + |D - G|, \end{aligned} \tag{3.4}$$

where A, B, C, D, E, F and G are defined in Figure 3.4.

The predictor $\hat{x}(u, v)$ is then obtained by the following:

if($d_v - d_h > 80$) /* *sharp horizontal edge* */

$\hat{x}(u, v) = A$

else if($d_v - d_h < -80$) /* *sharp vertical edge* */

$\hat{x}(u, v) = B$

else{

$\hat{x}(u, v) = (A + B)/2 + (D - C)/4$

if($d_v - d_h > 32$) /* *horizontal edge* */

$\hat{x}(u, v) = (\hat{x}(u, v) + A)/2$

else if($d_v - d_h > 8$) /* *weak horizontal edge* */

$\hat{x}(u, v) = (3\hat{x}(u, v) + A)/4$

else if($d_v - d_h < -32$) /* *vertical edge* */

$\hat{x}(u, v) = (\hat{x}(u, v) + B)/2$

else if($d_v - d_h < -8$) /* *weak vertical edge* */

$\hat{x}(u, v) = (3\hat{x}(u, v) + B)/4$

}

where the thresholds given are for 8 bits/pixel.

3.2.2 JPEG-LS

LOCO-I (Low Complexity, Context-Based, Lossless Image Compression) developed by Weinberger [65] [66] was adopted for the JPEG-LS standard for lossless and near-lossless coding of still images [8] [67]. JPEG-LS is organized into three fundamental steps: adaptive prediction, context modeling and Golomb-Rice coding. The first step is to predict a current pixel using surrounding pixels. A prediction $x(\hat{u}, v)$ of the current pixel $x(u, v)$ is obtained using pixels from previously encountered pixels $x^t = \{x_1, x_2, \ldots, x_{t-1}\}$. The set x^t is causal and finite. The set x^t consists of four pixels $\{A, B, C, D\}$ in the causal neighborhood of x_t as shown in Figure 3.4. The pixel index t will be dropped in the following when no confusion arises. The

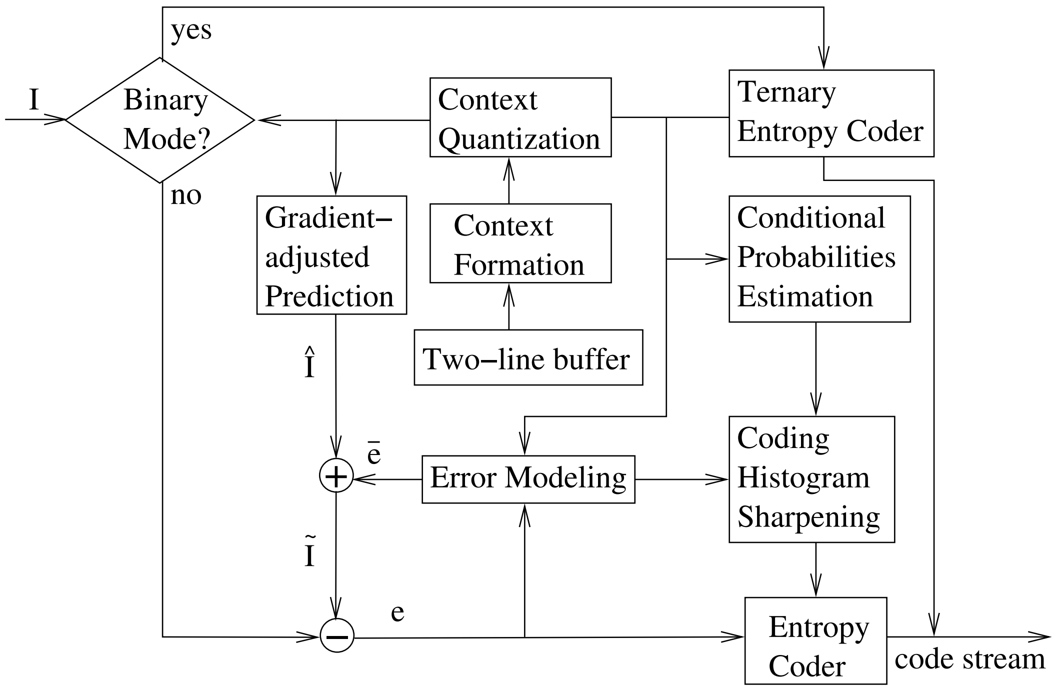


Fig. 3.3. Block Diagram of the CALIC Encoder.

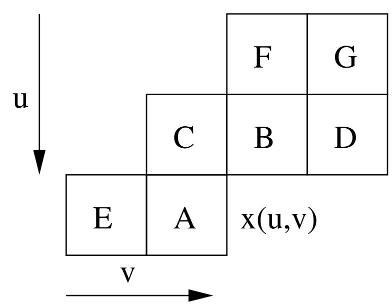


Fig. 3.4. Neighboring Pixels of the Current Pixel $x(u, v)$.

prediction is performed using a spatially variant combination of pixels in x^t. The predictor is known as MED (Median Edge Detector), which detects horizontal and vertical edges in the image:

$$\hat{x} = \begin{cases} min(A,B) & \text{if } C \geq max(A,B) \\ max(A,B) & \text{if } C \leq min(A,B) \\ A+B-C & \text{otherwise.} \end{cases} \tag{3.5}$$

The MED predictor always selects either the best, A or B, or the second-best, $A+B-C$, predictor among the three candidate predictors. The prediction residual e is obtained by subtracting the predicted value from the current value to be encoded:

$$e = x - \hat{x}. \tag{3.6}$$

The context that conditions the encoding of the current prediction residual is determined by quantized gradients [66]. The differences $g_1 = D - B$, $g_2 = B - C$ and $g_3 = C - A$ are quantized. These quantized differences represent the local gradient, that is, the level of activity of the neighbors, and are used to condition the statistical behavior of the prediction residuals. In JPEG-LS, the total number of different contexts is 365. For each context, the prediction error distribution is estimated using the prediction residuals.

The prediction residual e is an integer and is encoded with Golomb-Rice code using a context based probabilistic model. The range of e is $[-2^{b-1}, 2^{b-1}-1]$, where b is the number of bits to represent the prediction residual e. e typically has a two-sided geometric distribution, possibly with a nonzero average. The nonzero average can be estimated and removed via the so called *bias correction* procedure of JPEG-LS. In addition, the two-sided geometric distribution can be mapped into an one-sided distribution by using interleaving of positive and negative prediction residuals. The one-sided geometric distribution can be efficiently encoded by a Golomb-Rice code which is optimal for this distribution. JPEG-LS also includes a run-mode procedure to encode flat regions with constant pixel values using a run-length code. Near-lossless compression is also achieved by allowing a fixed maximum pixel error in the decompressed image. In this thesis, the run-mode and near-lossless compression will not be considered.

JPEG-LS treats color components almost independently. When JPEG-LS is used for a color image, better compression results are generally obtained, if inter-color spectral redundancies are removed [68] [69] [70]. However, such spectral decorrelation transform is not part of the standard and is best used at the application level due to the wide variety of color representations used in practice.

3.2.3 Improvements of JPEG-LS

The MED can detect horizontal and vertical edges reasonably well. However, its performance is not adequate for diagonal edges. Jiang [71] proposed a way to improve the diagonal edge detection. His method examines the local gradients of the neighbor pixels and estimates the existence of a diagonal edge. If a diagonal edge exists, the prediction is determined using a different method. The following is the revised MED prediction:

$$
\begin{array}{l}
if(C - \max(A,B) > Threshold_1)\ \{ \\
\quad if(A - B \geq Threshold_2)\ \{ \\
\qquad then\ \hat{x}(u,v) = \frac{D+B}{2} \\
\quad else\ \hat{x}(u,v) = \min(A,B) \\
\quad \} \\
\} \\
else\ if(C - \min(A,B) \leq Threshold_1)\ \{ \\
\quad if(A - B \geq Threshold_2)\ \{ \\
\qquad then\ \hat{x}(u,v) = \frac{D+\max(A,B)}{2} \\
\quad else\ \hat{x}(u,v) = \max(A,B) \\
\quad \} \\
\} \\
else\ \hat{x}(u,v) = A + B - C
\end{array}
\tag{3.7}
$$

where $\hat{x}(u,v)$ is the prediction of the current value $x(u,v)$. The result shows that the MSE (Mean Square Error) of the prediction residual image obtained by the proposed algorithm is marginally better than that of the conventional MED predictor.

Another attempt to improve the prediction of diagonal edges is reported by Edirisinghe [72]. The following is his modified MED prediction:

$$
\begin{aligned}
& if(C \geq \max(A,B))\ \{ \\
& \quad if((C - \max(A,B)) > T_1\ AND\ |A-B| \leq T_2)\ \{ \\
& \quad\quad then\ \hat{x}(u,v) = B + D - A \\
& \quad else\ \hat{x}(u,v) = \min(A,B) \\
& \quad \} \\
& \}
\end{aligned}
\tag{3.8}
$$

$$
\begin{aligned}
& else\ if(C \leq \min(A,B))\ \{ \\
& \quad if((\min(A,B) - C) > T_1\ AND\ (|A-B| \leq T_2)\ \{ \\
& \quad\quad then\ \hat{x}(u,v) = B + D - A \\
& \quad else\ \hat{x}(u,v) = \max(A,B) \\
& \quad \} \\
& \} \\
& else\ \hat{x}(u,v) = A + B - C.
\end{aligned}
$$

The result shows that the MSE (Mean Square Error) of the proposed algorithm is up to 8.51% better than that of the MED predictor proposed in JPEG-LS.

3.3 Lossless Image Compression Based on Spectral Decorrelation

If an image is in RGB color space, there are redundancies among the three color components, this is known as spectral redundancy. Penrose [9] uses a linear combination of neighboring pixels from the current color component and from a reference

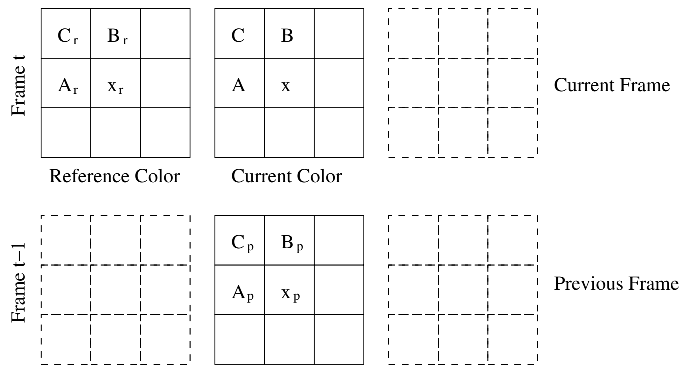


Fig. 3.5. Labeling of Spatial and Spectral Locations Near Pixel x.

color component to predict the current pixel. Figure 3.5 shows the labeling of spatial and spectral locations near pixel x. The spectral prediction $\hat{x}$ is the following:

$$\hat{x} = \frac{A + (x_r - A_r) + B + (x_r - B_r)}{2} \tag{3.9}$$

The gradients in the current color component are modeled by the horizontal and vertical intensity gradients in the reference color component.

Barequet's SICLIC (A Simple Inter-Color Lossless Image Coder) [73] is an inter-color coding algorithm based on a LOCO-like algorithm. While encoding the green (G) color component is done using MED prediction, encoding of the red (R) and blue (B) color components is based on spatial decorrelations as well as inter-color decorrelations. Assume that $Y(u, v)$ is the R component and $X(u, v)$ is the reference G component. Since $X(u, v)$ is known, the difference $D_k = X(u, v)$ - $Y(u, v), k = 1, 2, 3$ can be obtained. The MED may be used on D_k and not on R as in Figure 3.6. Prediction of the B component is a combination of predictions using the R component or the G component as a reference. Given the spatial predictor and the inter-color predictor, SICLIC selects the best: the predictor with lower average residuals.

An inter-color version of CALIC was proposed by Wu [68]. The inter-color CALIC exploits both inter-color and intra-color component statistical redundancies and ob-

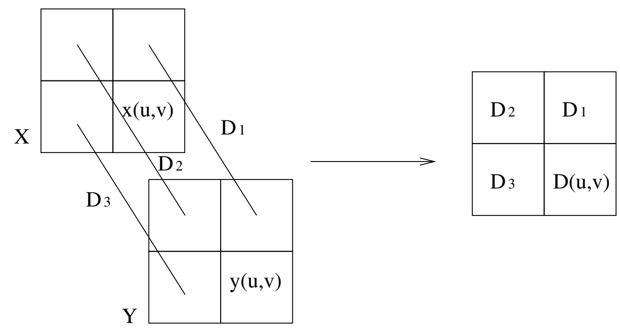


Fig. 3.6. Inter-Color Differences.

tains significant compression gains over its intra-color counterpart. The degree of correlation between color components can vary drastically from region to region. Thus, a correlation-based switched inter-color/intra-color predictor is used. Let $X = (x_1, x_2, \ldots, x_8)$ and $Y = (y_1, y_2, \ldots, y_8)$ be two random vectors whose elements are causal neighbors of the current pixel y in the reference and current color components respectively. The pixels x_i and y_i belong to the same spatial location but in different color components. If the correlation coefficient $\rho(X, Y)$ is greater than 0.5, then inter-color prediction is used. Otherwise, intra-color prediction is performed using GAP (Gradient Adjusted Predictor) as in CALIC [7]. For the inter-color prediction, the following is used:

$$\begin{aligned}
&If(|x - x_2| - |x - x_1| > T)\ \hat{y} = \hat{y}_h;\ /*\ sharp\ horizontal\ edge\ */\\
&Else\ if(|x - x_2| - |x - x_1| < -T)\ \hat{y} = \hat{y}_v;\ /*\ sharp\ vertical\ edge\ */\\
&Else\ \hat{y} = (\hat{y}_h + \hat{y}_v)/2
\end{aligned} \tag{3.10}$$

where $\hat{y}_h$ and $\hat{y}_v$ are extrapolations based on the horizontal and vertical gradients in the reference color component. On some multispectral images, inter-color CALIC has a reduction of more than 20% in data rate as compared to intra-color CALIC.

Carotti [10] decorrelates color components using a differential coding scheme, where two of the colors are represented by differences with the reference color. For

RGB, the green (G) color component is the reference color. The residual color components after the decorrelation are the following:

$$\begin{aligned}\hat{e}_g &= e_g \\ \hat{e}_r &= e_g - e_r \\ \hat{e}_b &= e_g - e_b\end{aligned} \tag{3.11}$$

where e_r, e_g and e_b is the residual after spatial and temporal prediction of the R, G and B color components, respectively.

Assche [74] proposes a novel color decorrelation technique for pre-press images. The Karhunen-Loeve Transform (KLT) is used to remove redundancies in the color components of each pixel. The KLT is used on small image regions with a more or less homogeneous color. The image is divided into equally-sized blocks or by segmentation. Spatial redundancies are handled by means of a simple linear predictor. The result shows that the color transform saves about a $\frac{1}{2}$ to two bits per pixel, compared to a purely predictive scheme.

3.4 Lossless Video Compression Based on Spatial Prediction

Spatial correlation between neighboring pixels can be removed by spatial prediction, such as the MED [8] or GAP [7] prediction techniques. Spectral correlations between color components can be reduced by spectral prediction or spectral decorrelation, such as a linear prediction or a reversible color transform. In a video sequence, there is one more correlation to be exploited. It is the temporal correlation between adjacent frames. Motion compensation using motion vectors is a temporal decorrelation technique. In lossy video compression motion compensation provides excellent performance. However, in lossless video compression, schemes utilizing only temporal correlations do not perform significantly better than schemes utilizing only spectral correlations [11]. Hence hybrid schemes that select the decorrelation technique adaptively among spatial, spectral and temporal predictions usually produce better results. Memon [11] investigated lossless compression of video sequences. In the paper, in order to incorporate temporal prediction, 3D versions of the lossless JPEG predictor (Table 3.1) are used. The following is an example of the 3D predictor from mode 4 of lossless JPEG:

$$\hat{x}(i,j) = \frac{2(x_k(i,j-1) + x_k(i-1,j) + x_{k-1}(i,j))}{3} - \frac{x_{k-1}(i-1,j) + x_{k-1}(i,j-1) + x_k(i-1,j-1)}{3} \tag{3.12}$$

Another temporal prediction method is to use block-based motion compensation. The differences between the current block and the best matched block in the previous frame, along with the displacement vector are encoded losslessly. In order to exploit spectral correlations, the predictor obtained from the R color component is used on the G color component and the predictor obtained from G color component is used on the B color component. The R color component can be either modeled by one of the spatial prediction models or by using the R color component of the previous frame for selecting a predictor. In order to maximize the performance, a hybrid scheme is used, which selects the one that has the smallest absolute sum of the prediction errors between spectral and motion compensated temporal prediction.

Penrose [9] proposed a lossless compression scheme for color video that took advantage of the spatial, spectral and temporal redundancy. For the spectral prediction, the prediction is performed as in Equation (3.9). For temporal prediction, it uses a simple prediction scheme in which x is predicted by the pixel at the same location in the previous frame as in Figure 3.5. A predictor which works well for some parts of the image may not work so well on other areas. Temporal prediction may work well for a static region in a video sequence, whereas spectral prediction may be more effective in areas with motion. The actual number of bits used in coding for each prediction mode is obtained and the best predictor with the minimum bits is selected. Each frame in the video sequence is divided into 5×5 blocks. Four predictors are available: spatial, spectral (using either red (R) or green (G) as the reference color component) and temporal. Block based adaptation is used for predictor selection and for the choice of the Golomb-Rice code parameter. The result shows the compression performance is better than that of lossless JPEG and JPEG-LS.

Carotti [10] presented backward-adaptive lossless compression of video sequences. Key features of the paper are a pixel-neighborhood backward-adaptive temporal predictor, an intra-frame spatial predictor and a differential coding scheme of the spectral components. The color of each pixel based on the color located in the same neighborhood in the previous two frames is predicted. The difference matrix is:

$$\Delta = \begin{pmatrix} \Delta_{-1,-1} & \Delta_{0,-1} & \Delta_{+1,-1} \\ \Delta_{-1,0} & \Delta_{0,0} & \Delta_{+1,0} \\ \Delta_{-1,+1} & \Delta_{0,+1} & \Delta_{+1,+1} \end{pmatrix} \tag{3.13}$$

where $\Delta_{i,j} = |p_{k-1}(x,y) - p_{k-2}(x+i, y+j)|$ and k is the frame number. The average value of the matrix, Δ_{avg}, and the sum of the matrix, Δ_{sum}, are obtained. The prediction is based on the pixels for which the differences are lower than Δ_{avg}. The prediction corresponds to a weighted average of the selected pixels using the weights

$$w_{i,j} = \frac{\Delta_{i,j}}{\Delta_{sum}} \tag{3.14}$$

The final prediction is:

$$\tilde{p_i}(x,y) = \sum_{j,k|\Delta_{j,k} \leq \Delta_{avg}} w_{j,k}\ p_{i-1}(x+j, y+k) \tag{3.15}$$

with $j, k \in [-1, 0, +1]$. For spatial prediction, the MED predictor in JPEG-LS is used. To take advantage of both spatial and temporal prediction, the average of the two predictions is formed:

$$\bar{p_i}(x,y) = a\tilde{p_i}(x,y) + b\hat{p_i}(x,y) \tag{3.16}$$

where $\hat{p_i}(x,y)$ is the spatial prediction of the current pixel x and a, b are the weights. The spectral decorrelation is used as in Equation (3.11). The paper shows that the performance is up to 12% better than that of the LOCO-I [66] in terms of compression ratio.

Brunello [75] [76] proposed a lossless video compression technique which is based on motion compensation, optimal 3D linear prediction and context based Golomb-Rice entropy coding. First, block based motion compensation is performed. The motion vectors relative to each block are determined and each pixel is predicted using a linear combination of pixels in the current and previous frames, by taking into account motion information. The optimal linear prediction coefficients are adaptively formed for each pixel. The compression gain of 0.8 bit/pixel is achieved with respect to static JPEG-LS.

4. ADAPTIVE LOSSLESS VIDEO COMPRESSION

Figure 4.1 shows the block diagram of the new proposed video encoder. Every frame of the color video sequence first undergoes a RCT (reversible color transform) to exploit spectral redundancy if the video sequence is in the RGB color space. In the "Integer Wavelet Transform Selector" block, the most effective IWT (integer wavelet transform) for each color component is determined. One level of the selected IWT is performed on the corresponding color component in the "Integer Wavelet Transform" block. The 3 selected IWT IDs for a frame are transmitted to the decoder. Each subband is partitioned into $N \times N$ blocks. Then the MED (Median Edge Detector) spatial prediction and *backward-adaptive* temporal prediction are performed on each block. A frame buffer is used to store the previous frame. $e1$ and $e2$ are the residuals from the spatial prediction and the backward-adaptive temporal prediction, respectively. In the "Block-Based Adaptive Selector" block, the prediction mode between spatial prediction and temporal prediction for each block is selected

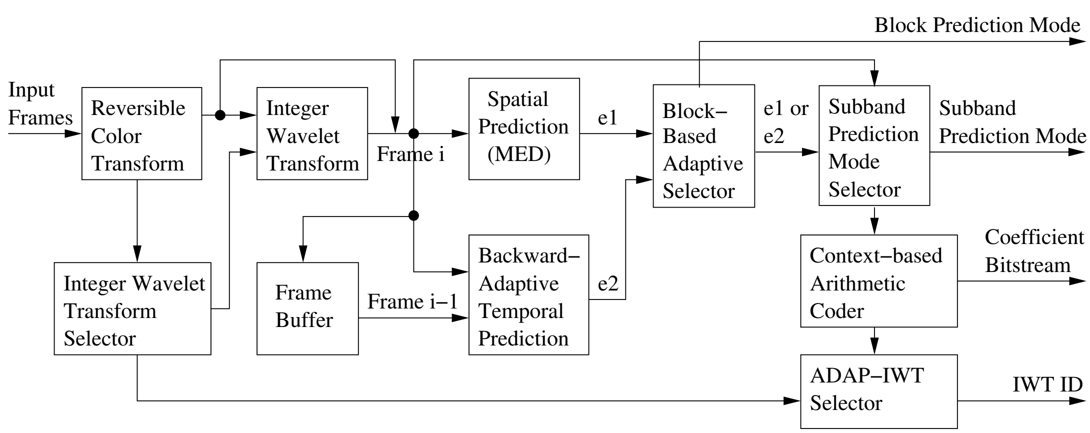


Fig. 4.1. Block Diagram of the Proposed Video Encoder.

using the SAD (Sum of Absolute Difference). This is known as "adaptive selection of spatial and temporal prediction." The block prediction mode is then transmitted to the decoder. No side information, for example motion vectors, is transmitted for the temporal prediction.

In the "Subband Prediction Mode Selector" block, the subband prediction mode is determined using the entropy of the residuals obtained by the adaptive selection of spatial and temporal prediction and the entropy of the wavelet coefficients. The subband prediction mode is also transmitted to the decoder. If the entropy of the wavelet coefficients is smaller than that of the residual error, we label this as the *Direct Sending* mode. In the *Direct Sending* mode, no information concerning the block prediction mode and the residual error is transmitted. Instead, the wavelet coefficients of the subband are encoded. The other case is known as *Prediction* mode. In the *Prediction* mode, the residual error and the prediction mode for each block are transmitted. The above encoding procedure is repeated without using the IWT. In this case, "Subband" is replace by "Frame." In the "Adap-IWT Selector" block, the best approach between using IWT or not using IWT is determined. The "Context-Based Arithmetic Coder" block is used to encode the residual error or the wavelet coefficients according to the subband prediction mode.

At the decoder, the same procedure is performed in the reverse order to recover the original pixels. The decoding process can be done much fast than the encoding process since each operation mode is included in the bitstream.

The proposed video encoder uses three adaptation methods to improve the compression performance and the side information for these modes is transmitted to the decoder. The side information at the color component level is the type of IWT (*IWT ID*) used in the color component. One *IWT ID* is sent to the decoder for each color component. The side information at the subband level is the subband prediction mode, *Prediction* and *Direct Sending* mode. Four prediction mode indicators are needed for each color component. The side information at the macro block level is the macro block prediction mode, *spatial* and *temporal* prediction. Prediction mode

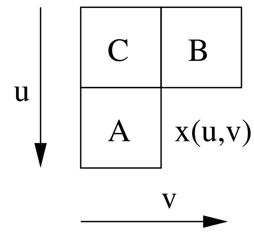


Fig. 4.2. The neighboring pixels of $x(u, v)$ used for spatial prediction.

indicator for each macro block should be sent to the decoder. The search range for temporal prediction is sent for each frame. The frame size is sent one time at the beginning of encoding a video sequence.

4.1 Spectral Decorrelation (Reversible Color Transform)

If a video sequence is in the RGB color space it is known that spectral redundancy exists among the three color components. Video sequences usually are transformed into a luminance/chrominance color space such as the YUV or YCrCb color space to exploit spectral redundancy. The luminance/chrominance transformations used in MPEG typically are not reversible. Since here we are interested in lossless compression the RCT (Reversibel Color Transform) color space as described in JPEG-2000 [2] [60] is used. The forward and backward RCT transform is described by Equations 3.2 and 3.3 in Section 3.1.1 respectively.

4.2 Spatial Prediction

In order to remove spatial redundancy in a frame, predictive coding (DPCM) is used whereby an estimate of a pixel is obtained and the estimation error is then encoded. The efficiency of the method is determined by the type of predictor. The MED (Median Edge Detector) predictor as described by JPEG-LS [8] is used. MED can detect horizontal and vertical edges by examining the left (A), the upper (B) and

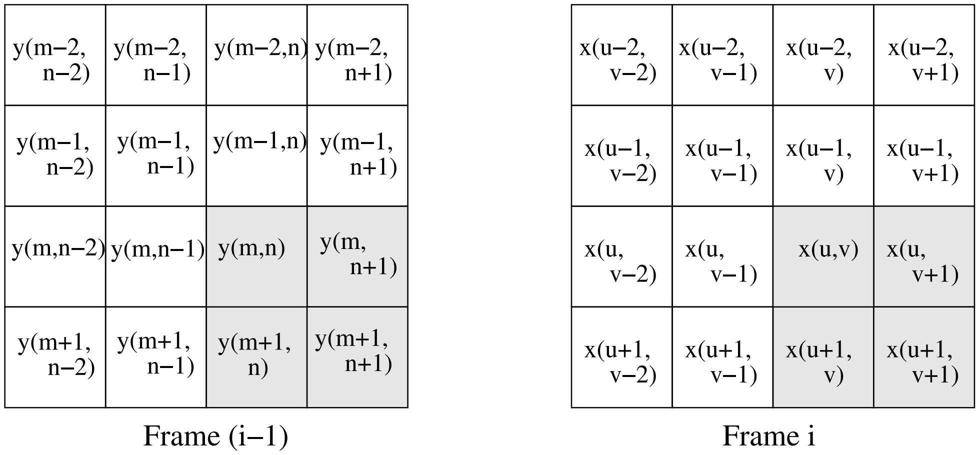


Fig. 4.3. The neighboring pixels of $x(u,v)$ used for temporal prediction.

the upper-left (C) neighbors of the current pixel $x(u,v)$ as shown in Figure 4.2. This approach has been shown to be very robust across many different types of images. The prediction is performed according to the following:

$$\hat{x}(u,v) = \begin{cases} min(A,B) & \text{if } C \geq max(A,B) \\ max(A,B) & \text{if } C \leq min(A,B) \\ A+B-C & \text{otherwise} \end{cases} \tag{4.1}$$

where $\hat{x}(u,v)$ is the prediction of $x(u,v)$. The MED predictor always selects either the best or the second-best predictor among the three candidate predictors.

4.3 Backward-Adaptive Temporal Prediction

Temporal prediction removes redundancies between frames. In lossy compression, block based temporal prediction by motion compensation provides very high coding gain. Motion vectors, needed to reconstruct the frame, are transmitted as overhead data. If the block size for motion estimation is large, then the total number of the motion vectors for a frame is small. Thus, a small number of bits is required for encoding the motion vectors; however, the entropy of the motion compensated residual frame may be large. If the block size for motion estimation decreases,

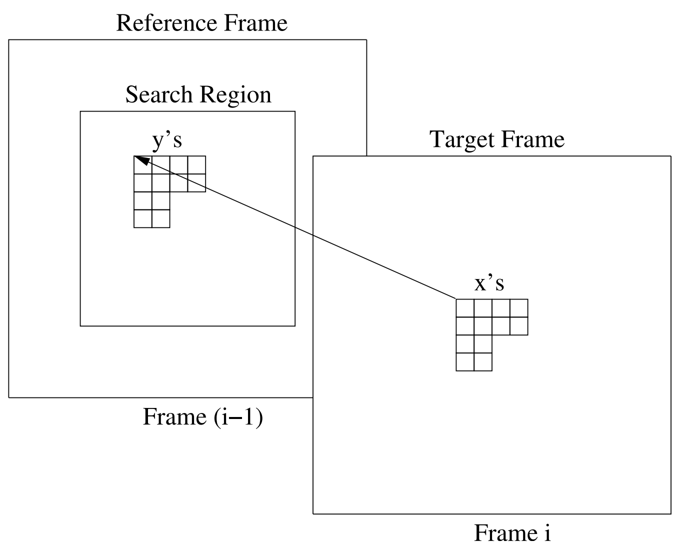


Fig. 4.4. Block Search used for Temporal Prediction.

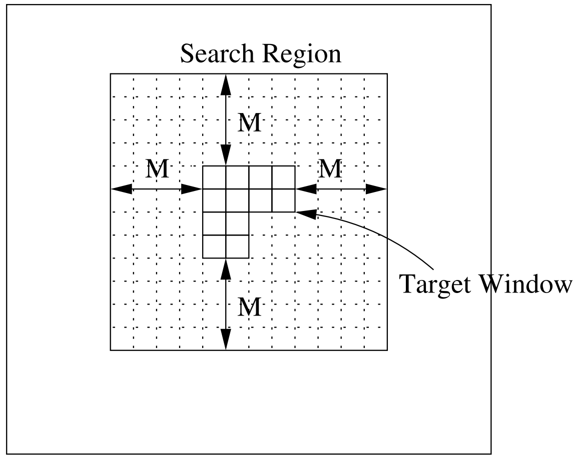


Fig. 4.5. Search Region Defined by the Search Range M.

then the entropy of the motion compensated residual frame decreases; however, the required bits for encoding the motion vectors increase. The choice of the block size for motion estimation has a significant impact on the compression performance. In general the block size cannot be very small.

Motion information is used in a slightly different manner for our new method. To predict the motion from Frame $i-1$ to Frame i we divide the subband to be predicted into non-overlapping 2×2 blocks. In Figure 4.3 the shaded 2×2 block in Frame i is the block needed to be encoded using motion prediction based on knowing the previous Frame $i-1$ and other blocks in Frame i. This block will be called the "current block." The region used for the prediction consists of all the pixels except the shaded pixels of Frame i shown in Figure 4.3. This is denoted as the "target window." Thus the target window is a causal neighborhood of the current block. The pixels of the target window are known to both the encoder and decoder.

The motion search is performed as shown in Figure 4.4. The goal is to match the current target window from Frame i with a similarly defined target window in Frame $i-1$. For the work reported here; the search region for the best match is regions defined by the search ranges 0, 2, 4 and 6 pixels that bound the target window in Frame i. The search region defined by a search range M is shown in Figure 4.5. The SAD (Sum of Absolute Difference) is obtained between the current target window from Frame i and the reference target window from Frame $i-1$ for all the blocks in the search region as shown in Figure 4.4. Specifically for the current block $\{x(u+s,v+t)|s,t \in [0,1]\}$, the SAD is defined by the following:

$$SAD_{uv}(m,n) = \sum_{i=-1}^{-2}\sum_{j=-2}^{1} |x(u+i,v+j) - y(m+i,n+j)| + \tag{4.2}$$

$$\sum_{i=0}^{1}\sum_{j=-2}^{-1} |x(u+i,v+j) - y(m+i,n+j)|,$$

where m $\in$ [u-M, u+M], n $\in$ [v-M, v+M] and M is the search range.

Indices k and l with the minimum SAD are obtained by the following:

$$k = argmin_m SAD_{uv}(m,n) \tag{4.3}$$

$$l = argmin_n SAD_{uv}(m,n).$$

The prediction, $\hat{x}$, of the current block is then determined by the target window with the smallest error. Therefore the predicted block for the current block is the following:

$$\hat{x}(u+s,v+t) = y(k+s,l+t), \qquad s,t \in [0,1] \tag{4.4}$$

The basic assumption is that if blocks in the target window have similar values between Frame i and Frame $i-1$ then their corresponding current block also has similar values as shown in Figure 4.3. If the SAD of the non-shaded pixels between Frame i and Frame $i-1$ is minimal in the search area, then the shaded pixels in Frame i may be predicted by the shaded pixels in Frame $i-1$.

The residual error is obtained by the following:

$$e(u+s,v+t) = x(u+s,v+t) - \hat{x}(u+s,v+t), \tag{4.5}$$

where s,t $\in$ [0,1]. This procedure is repeated for all blocks. The above procedure can be used at the decoder in the exact same way. Thus, this is known as *backward-adaptive* temporal prediction.

4.4 Adaptive Prediction Mode Selection

If there is a great deal of motion in the video sequence, temporal prediction does not perform well with respect to compression efficiency (data rate). In this case, only spatial prediction is used. In other cases, temporal prediction may work better than spatial prediction. The decision whether spatial or temporal prediction is used can be made on a block-by-block basis. For a given block, each prediction technique is used (spatial and temporal) and the SAD between the current and the predicted block is evaluated. The prediction technique with the smaller SAD is selected as the prediction technique for that block and the prediction mode is transmitted to the decoder as side information. One bit for each block is needed for the "prediction-type" flag. The overhead of sending the side information should be small. If the block size is 16×16 the entropy reduction by adaptive selection of the prediction mode should compensate for the overhead.

If an IWT is used, the above adaptive prediction mode is done on each subband in the wavelet domain. In this case, another adaptation step is needed to decide whether the original wavelet coefficients or the prediction residuals are encoded for each subband. After finding one level of the IWT, the subbands LH, HL and HH contain high frequency components of the original signal. There may not exist much spatial correlation between adjacent coefficients in the high frequency subbands. There may not also exist temporal correlation between corresponding subbands of successive frames. In these cases, the entropy of the adaptive prediction mode may be larger than that of the wavelet coefficients. If the entropy of the original wavelet coefficients is smaller than that of the prediction residuals obtained by the adaptive prediction mode, the wavelet coefficients are encoded and transmitted to the decoder. This is known as the *Direct Sending* mode. The prediction residuals and side information are encoded and transmitted, otherwise. This is known as the *Prediction* mode. Typically the LL subband uses the *Prediction* mode and the HL, LH and HH subbands use the *Direct Sending* mode. This information is also transmitted to the

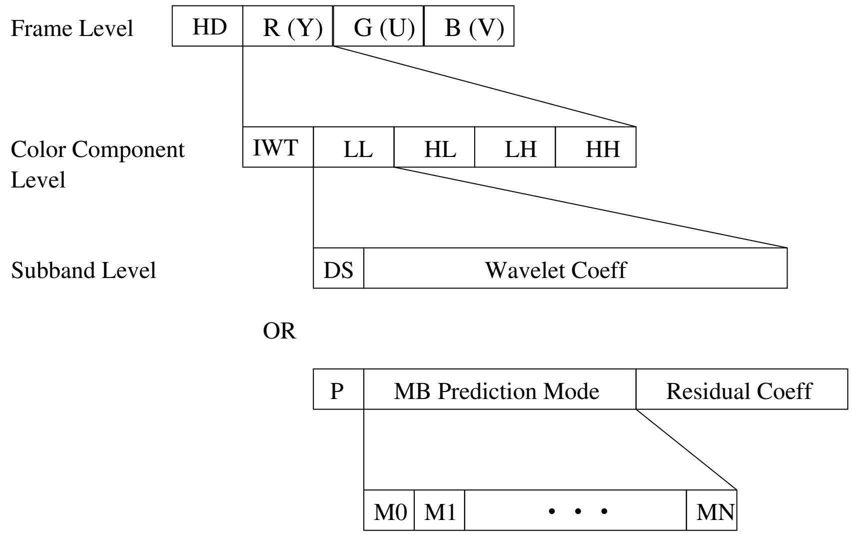


Fig. 4.6. Bitstream Structure for a Color Frame: HD: Frame Header, IWT: IWT ID, P: *Prediction* mode for the Subband, DS: *Direct Sending* mode for the Subband, MB Prediction Mode: Prediction Mode for the Macro Blocks, M0, ..., MN.

decoder. At the decoder, the subband set using the *Direct Sending* mode can be decoded very fast since there is no need to perform the adaptive prediction.

Figure 4.6 shows the compressed bitstream structure for a frame when an IWT is used. The bitstream consists of 3 color components in "Frame level." There are 4 subband parts in the "Color component level." In the "Subband level," there are two cases. One case is that the subband is selected for the *Prediction* mode. In this case, the prediction mode for each macro block and the prediction residuals of the subband are transmitted. The other case is when the *Direct Sending* mode is selected for the subband. In this case, only the wavelet coefficients of the subband are transmitted. Figure 4.7 shows an example of the selected prediction mode for a 128×128 frame (one color component). The prediction mode for the LL subband is the *Prediction* mode and the *Direct Sending* mode for the other 3 subbands. Thus, the bitstream at the "Color component level" is shown in Figure 4.8. If an IWT is

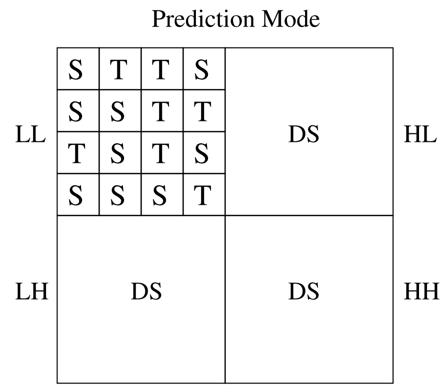


Fig. 4.7. Example of Prediction Modes for a Color Component with Frame Size 128 × 128: S: Spatial Prediction for the Macro Block, T: Temporal Prediction for the Macro Block, DS: *Direct Sending* mode for the subband.

not used, the bottom part of the "Subband level" becomes the bitstream structure of the "Color component level."

Table 4.1 shows the side information to be sent to the decoder for a color component. The side information at the color component level is the type of IWT (*IWT ID*) used in the color component. If *IWT ID* = 0, then no IWT is used for the color component. Four IWTs are used: the S Transform in Equation (2.16), the (2,2) Transform in Equation (2.20), the (2+2,2) Transform in Equation (2.21) and the (4,4) Transform in Equation (2.22). One *IWT ID* is sent to the decoder for each color component. The side information at the subband level is the subband prediction mode, *Prediction* and *Direct Sending* mode. Four prediction mode indicators are needed for each color component since one level of the IWT is used. The side information at the macro block level is the macro block prediction mode, *spatial* and *temporal* prediction. The total number of prediction mode indicator at the macro block level for each color component is $MN/(16{\times}16)$, where M and N are the width and height of the frame respectively. The search range for temporal prediction is sent for each frame. The frame size is sent one time at the beginning of encoding a video sequence.

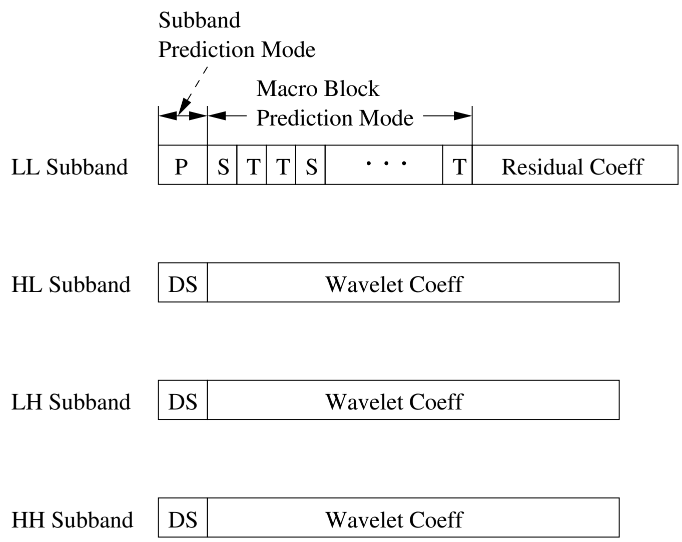


Fig. 4.8. Bitstream at the Color Component Level for Figure 4.7.

Table 4.1
Side Information for a Color Component.

Side Information	Description
Color Component IWT ID	0: No IWT
	1: S Transform
	2: (2,2) Transform
	3: (2+2,2) Transform
	4: (4,4) Transform
Subband Prediction Mode	0: Direct Sending
	1: Prediction
Macroblock Prediction Mode	0: Spatial Prediction
	1: Temporal Prediction
Search Range	For temporal prediction

Table 4.2
Residual Entropy (Bits/pixel) for Adaptive Selection of Spatial and Temporal Prediction with respect to the Search Range M.

Sequence		MED	Search Range M			
			0	2	4	6
RGB	Claire	6.735	**6.345**	6.426	6.461	6.478
	Football	11.299	11.230	11.197	11.184	**11.180**
	Miss America	8.935	8.867	**8.624**	8.649	8.652
	Mobile-Calendar	12.427	12.398	**11.898**	11.979	12.032
YUV 720×480	Flowergarden	8.069	8.057	7.794	7.152	**6.928**
	Football	6.903	6.734	6.669	**6.656**	6.658
	Hockey	4.116	4.119	4.119	4.117	**4.108**
	Mobile-Calendar	8.173	8.070	**6.944**	7.001	7.046
	Tennis	7.221	6.355	6.307	**6.222**	6.260

4.5 Search Range for Backward-Adaptive Temporal Prediction

During motion estimation, a search range is needed to confine the search area. The search range affects the performance of the motion estimation. If the search range is large, then more accurate motion vectors can be obtained, however this requires more search time. In conventional motion estimation, the current block is used to find a motion vector. The best matched block in the search range is selected and the motion vector is computed. Typically a larger search area provides opportunities to find the better matched blocks. However, in backward-adaptive temporal prediction, the current block is not used for the search. Instead, the current block's neighboring blocks, the *target windows*, are used. In this case, a larger search area does not guarantee better results as shown in Table 4.2. The search area defined by the search range M is defined in Figure 4.5. In Table 4.2, the entropy in bits/pixel is obtained for search range M from 0 to 6 with the bold number in

Table 4.3
Residual Entropy (Bits/pixel) for Adaptive Selection of Spatial and Temporal Prediction using the IWT with respect to the Search Range M.

Sequence		Search Range M 0	2	4	6
RGB	Claire	**5.766**	5.953	6.010	6.036
	Football	9.443	**9.440**	9.473	9.498
	Miss America	**7.617**	7.643	7.618	7.624
	Mobile-Calendar	10.902	**10.761**	10.815	10.847
YUV 720×480	Flowergarden	7.709	7.349	**7.082**	7.087
	Football	6.520	**6.484**	6.491	6.504
	Hockey	3.780	3.778	3.771	**3.768**
	Mobile-Calendar	7.910	**7.655**	7.708	7.738
	Tennis	6.405	**6.324**	6.373	6.416

each row representing the best result for the sequence. The best search range is different for each video sequence. The amount of motion in a sequence roughly determines the best search range. For example, the "Football" in the RGB color space, "Flowergarden" and "Hockey" contain relatively high motion content. The best search range for these sequences is 6. However, the "Mobile-Calendar" and "Miss America" have relatively less motion with the best search range 2. The best search range for the "Claire" is 0.

If an IWT is used before spatial and temporal prediction, the LL subband typically uses the *Prediction* mode and the other subbands use the *Direct Sending* mode. The subbands in the *Prediction* mode only affect the best search range since temporal prediction is used for the subbands in the *Direct Sending* mode. The best search range for the subbands in *Prediction* mode become smaller due to the subsampling of the IWT. Table 4.3 shows the best search range when an IWT is used. The best search ranges in Table 4.3 tend to be reduced when compared with the results in Table 4.2. For example, the best search range for the “Miss America” sequence is reduced to 0 from 2 in Table 4.2. For the “Tennis” sequence, it is reduced to 2 from 4. It is reduced to 4 from 6 for the “Flowergarden” sequence.

4.6 Entropy Coding of the Prediction Residuals

Error coefficients in the prediction residual frames are considered as an array of symbols to be encoded and transmitted. The residual frames still have textures with spatial dependencies that can be localized in limited neighborhood and described by a *context-based statistical model*. The purpose of context modeling is to obtain a skewed probability distribution of the symbols, which has smaller entropy. Suppose a context set $C = 0, 1, \ldots, N-1$ is given, where the context is generated by a context modeling function, which is a mapping of a context template T into a context value C, $F : T \rightarrow C$. For each symbol x to be encoded, a conditional probability mass function $p(x|c)$ is estimated by switching between different probability models according to the context $c \in C$. x is encoded using the estimated conditional probability mass function $p(x|c)$ where the probability model is updated with the value x. Thus the probability model is estimated by updating the model according to the actual source symbols. The context template used in our algorithm is shown in Figure 4.9. The context is determined by the quantization of the sum of the absolute error values in the context template:

$$\begin{aligned} z &= |E_a| + |E_b| + |E_c| \\ c &= Q(z), \end{aligned} \tag{4.6}$$

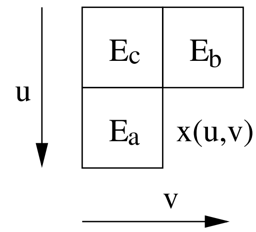


Fig. 4.9. The Context Template for Context Modeling.

where E_a, E_b and E_c are the residual errors and $z \in [0, 1, \ldots, 765]$. z is then quantized into 8 regions by a quantizer $Q(\cdot)$. The quantization is described by the following:

$$\begin{aligned}
&if(z < Q_0)\ then\ c = 0 \\
&else\ if(Q_0 \leq z < Q_1)\ then\ c = 1 \\
&else\ if(Q_1 \leq z < Q_2)\ then\ c = 2 \\
&else\ if(Q_2 \leq z < Q_3)\ then\ c = 3 \\
&else\ if(Q_3 \leq z < Q_4)\ then\ c = 4 \\
&else\ if(Q_4 \leq z < Q_5)\ then\ c = 5 \\
&else\ if(Q_5 \leq z < Q_6)\ then\ c = 6 \\
&else\ c = 7,
\end{aligned} \tag{4.7}$$

where $Q_i, 0 \leq i \leq 6$ are the quantization thresholds. The default values for the Q_i's are (2, 5, 8, 12, 18, 32, 64). These values are obtained by examining the histogram of the residual errors.

4.7 Adaptive IWT (Integer Wavelet Transform) Selection

In video sequences, consecutive frames have similar statistics; that is, they are highly correlated. However the statistics gradually changes in time, or abruptly change in the case of a scene change. After some time, the statistics of the frames can be different from the previous frames. When a wavelet transform is used in a video compression algorithm, the compression performance of the encoder depends on the wavelet transform used and also on the content of the frames in the video

Table 4.4
Data Rates (Bits/pixel) for IWTs with Search Range 2.

Sequence		IWT (Integer Wavelet Transform)			
		S Trans	(2,2) Trans	(2+2,2) Trans	(4,4) Trans
RGB	Football	**9.507**	12.842	12.792	13.888
	Mobile-Calendar	12.581	10.859	**10.747**	13.462
	Claire	**5.762**	7.600	7.657	8.282
	Miss America	**7.853**	10.344	10.343	11.848
YUV 720×480	Miss America	4.972	**4.707**	4.711	5.352
	Football	6.628	6.311	**6.274**	6.923
	Flowergarden	7.287	6.988	**6.894**	7.505
	Hockey	4.121	3.737	**3.682**	4.452
	Mobile-Calendar	7.433	**7.309**	7.317	7.640
	Tennis	6.172	**6.138**	6.171	6.217

sequence. For a given frame, different wavelet transforms show different compression performance. A wavelet transform having the best compression result for a frame does not guarantee that the wavelet transform works well for another frames. In Table 4.4, the compression performance of various IWTs (integer wavelet transform) is shown. The four IWTs are used: the S Transform in Equation (2.16), the (2,2) Transform in Equation (2.20), the (2+2,2) Transform in Equation (2.21) and the (4,4) Transform in Equation (2.22). In Table 4.4, the best data rate for video sequences is represented as bold number in each row. The S Transform shows the best compression result for the "Football," "Claire" and "Miss America" in the RGB sequence. The (2,2) Transform is the best for the "Miss America" in the YUV sequence, "Mobile-Calendar" and "Tennis." The (2+2,2) Transform is the best for the "Mobile-Calendar" in the RGB sequence, "Football" in the YUV sequence, "Flowergarden" and "Hockey."

The best wavelet transform is different for each video sequence shown in Table 4.4. The compression performance can be improved if the best wavelet transform is used for each frame or each color component. However, there is not an efficient and reliable method to determine an appropriate wavelet transform for a given frame or a color component.

Adams [19] shows that the smoothness and the uniformity of an image are important factors in determining the best IWT. The metrics for smoothness and uniformity are defined below.

Assume the range of pixels to be $0, 1, \ldots, R-1$. The differences between adjacent pixels in the horizontal and vertical directions, d_i, are defined for $i = 0, 1, \ldots, N-1$. The smoothness measure is defined as:

$$s = 100\frac{N_s}{N}, \tag{4.8}$$

where N_s is the number of d_i such that $|d_i| \geq \frac{R}{2}$ and $s \in [0, 100]$. Small s value corresponds to greater smoothness. The uniformity measure is similarly defined as:

$$u = 100\frac{N_u}{N}, \tag{4.9}$$

where N_u is the number of d_i such that $d_i = 0$. This measure also has values on the interval $[0, 100]$ with larger values corresponding to greater uniformity. For a given image, the smoothness metric s and the uniformity metric u are used to determine the best wavelet transform.

Saha [20] presented several image activity measures (IAM) that are defined by using image variance, edges, wavelet coefficients and gradients were used to predict compression performance. Saha shows that the gradient-base IAM, defined below, is the most effective measure in determining the compressibility of an image.

$$IAM = \frac{1}{MN}[\sum_{i=1}^{M-1}\sum_{j=1}^{N}|x[i,j]-x[i+1,j]| + \sum_{i=1}^{M}\sum_{j=1}^{N-1}|x[i,j]-x[i,j+1]|], \tag{4.10}$$

where M and N are the image width and height respectively. If the IAM is large, then the compression ratio of the image is smaller than that of other images for a

Table 4.5
IWT Selection Criteria Based on the Uniformity and the IAM.

Range of u and IAM	IWT Selection
$0 \leq u \leq 6$ and $25 \leq IAM$	(4,4) Transform
$u \geq 25$ and $0 \leq IAM \leq 2.5$	(4,4) Transform
$u \geq 30$ and $2.5 \leq IAM \leq 15$	S Transform
elsewhere	(2+2,2) or (2,2) Transform

given wavelet transform. If the IAM is small, then the compression ratio is larger than that of other images for the given wavelet transform.

We would like to find a method for determining the best wavelet transform for a given color component. We tested different combinations of the above metrics: the smoothness, the uniformity and the IAM. The combination of Adams' uniformity and Saha's gradient-based IAM is the most promising. Figure 4.10 shows the uniformity and IAM relationship. Video sequences in the RGB and YUV color space are used in this experiment. Twenty Sample frames from each video sequence were used and the uniformity and the IAM are computed for each color component. The data rate for each IWT is also obtained and the best IWT is determined for each color component. In Figure 4.10, the upper-far-left region and the bottom-right region are assigned to the (4,4) Transform, the upper-left region (right side of the (4,4) Transform region) is assigned to the S Transform and the other regions are assigned to the (2,2) and (2+2,2) Transforms. For the (2,2) and (2+2,2) Transforms, the result is mixed. Thus, it is not possible to classify the regions for the two transforms. However, both transforms show similar compression performance. From Figure 4.10, the criteria for the IWT selection is described in Table 4.5. Some points of the (2+2,2) Transform are located in the S Transform region and those points are assigned to the S transform by the above criteria. The criteria for the IWT selection can be used when the encoding complexity is important.

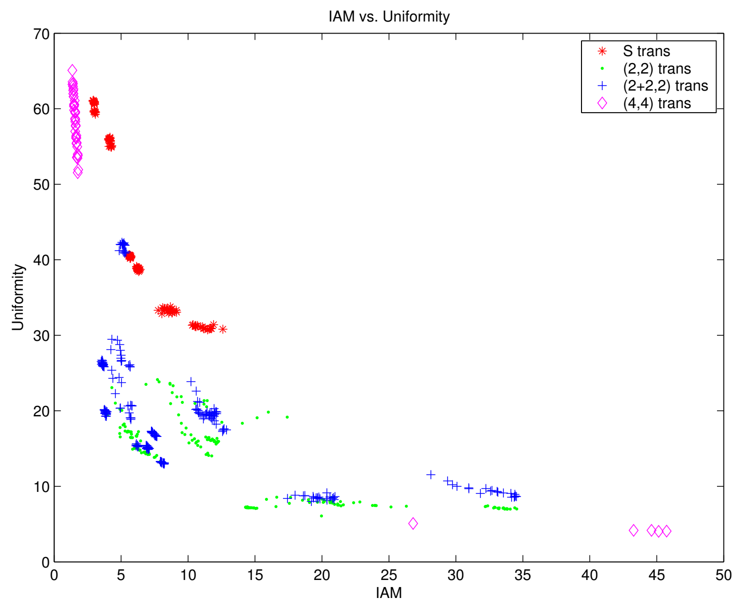


Fig. 4.10. IAM vs. Uniformity for Adaptive IWT Selection.

4.8 Video Compressibility

If the IAM is large, then there are large variations of pixels across the image, that is, there are many edges and complex structures in the image. If the IAM is small, then there are not many variations in the pixels across the image. That is, the spatial correlation is high, thus spatial prediction works well. Saha [20] showed that, for a given wavelet transform, an image with a smaller IAM was compressed better than an image with a larger IAM.

Similarly, a new measure can be defined for video compressibility. Assume $x[i, j]$ to be the current frame and $y[i, j]$ to be the previous frame. TIAM (Temporal Image Activity Measure) is defined as the following:

$$TIAM = \frac{1}{MN}[\sum_{i=1}^{M}\sum_{j=1}^{N}|x[i, j] - y[i, j]|], \tag{4.11}$$

where M and N are the frame width and height respectively. The TIAM estimates the inter-frame activity between two consecutive frames. Figure 4.11 shows the relationship between TIAM and the entropy of several compressed video sequences. Figure 4.11 shows that a video sequence with a large TIAM cannot be well compressed and a video sequence with a small TIAM can be compressed well. Although the "Fountain" sequence has a small TIAM, the compression result of the sequence is exceptionally good because the sequence is computer-generated and has large static backgrounds.

If the TIAM is large, there are large variations in the pixels in two successive frames, that is, a lot motion in the frames. In this case, temporal prediction does not contribute much to the compression performance, since there is not much correlation between two successive frames. If the TIAM is small, there are not many changes in the pixel values in two consecutive frames. In this case, temporal prediction techniques do contribute much to the compression performance.

In order to show the video compressibility with respect to TIAM, TIAM and IAM are used together. IAM and TIAM of the Y, U and V color components for each frame are obtained and plotted in Figure 4.12 through Figure 4.19. After examining the figures, a line is chosen to be used as a criteria for determining video compressibility. The line is defined as:

$$y = \frac{20}{25}x, \tag{4.12}$$

where x ∈ IAM and y ∈ TIAM. If $(x, y) = (IAM, TIAM)$ is above y, then the video compressibility of the frame is low. Thus, the contribution of temporal prediction may not be significant to the compression performance. If (x, y) is located farther above y, the contribution of temporal prediction becomes smaller. If (x, y) is below y, the contribution of temporal prediction is large. If (x, y) is located farther below y, the contribution of temporal prediction becomes larger. In this case, better compression of the frame can be achieved by incorporating temporal prediction techniques.

Figures 4.12 and 4.13 show the relationship between IAM and TIAM for the 176×144 YUV sequences. For the "Akiyo" sequence, IAM is concentrated near 9.7, 5.7 and 2.7 for the Y, U and V color components respectively and the TIAM is close to 0.2 for all three color components. The IAM is small for this sequence. However, the TIAM is much smaller. All points are below y. Thus temporal prediction will contribute much to the compression performance. For the "Carphone" and "Coastguard" sequences, many points are below y. Thus temporal prediction will also contribute much to the compression performance. For the "Glasgow" sequence, most of the points are below y. Thus, this sequence should also use temporal prediction.

Figures 4.14 and 4.15 show the relationship between IAM and TIAM for the 352×288 YUV sequences. Most points are below y, thus temporal prediction will contribute to the compression performance. The compression of the "Foreman," "Paris," and "Tempete" sequences will be improved by temporal prediction. However, for the "Bus" sequence, it is not clear that temporal prediction will contribute to the compression performance, since many points are close to y.

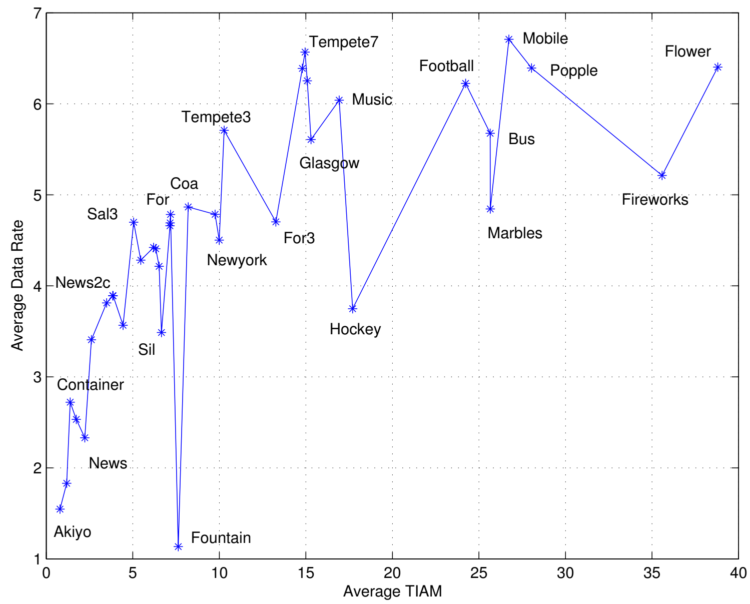


Fig. 4.11. Video Compressibility.

In Figures 4.16 and 4.17, the relationship between IAM and TIAM for the 720×480 YUV sequences shows that most points are below y. Thus temporal prediction will contribute to the compression performance.

Figures 4.18 and 4.19 show the relationship between IAM and TIAM for other 720×480 YUV sequences. Most points above y indicates the contribution of temporal predciton will not be large.

4.9 Experimental Results

The proposed lossless video encoder was tested on 32 color video sequences in the RGB and YUV color spaces. Some sequences have more than one frame size, thus, the total number of test sequences is 43. Figures 4.20, 4.21 and 4.22 show sample frames from each sequence. As a performance measure of the proposed video encoder, the data rates of the compressed video sequences after using an arithmetic encoder are obtained. In order to compare the performance of the proposed encoder,

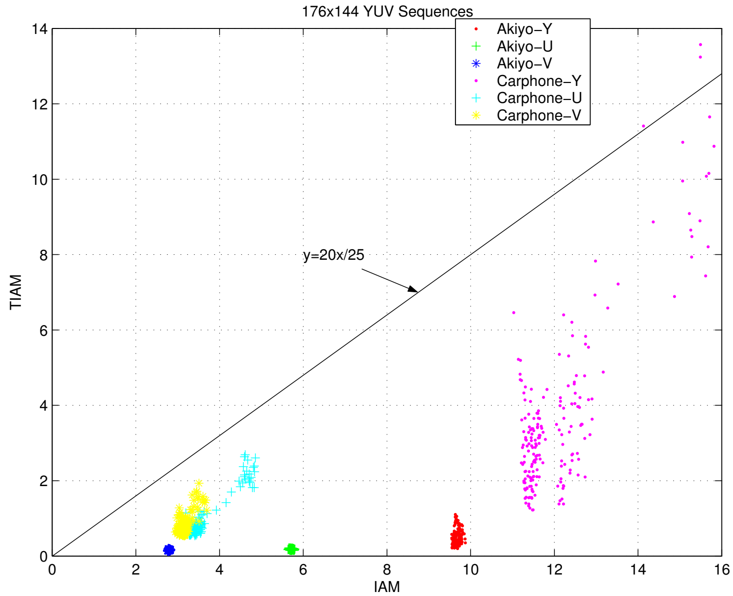


Fig. 4.12. Video Compressibility for the YUV 176×144 Akiyo and Carphone Sequences.

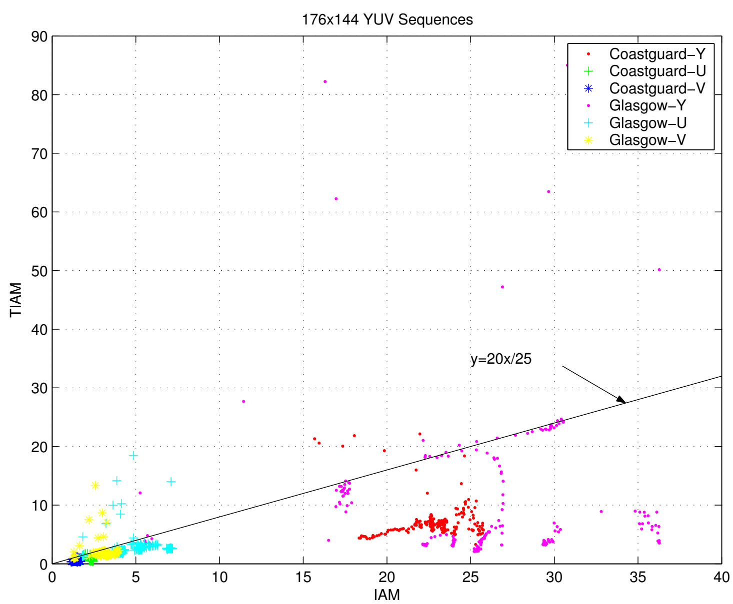


Fig. 4.13. Video Compressibility for the YUV 176×144 Coastguard and Glasgow Sequences.

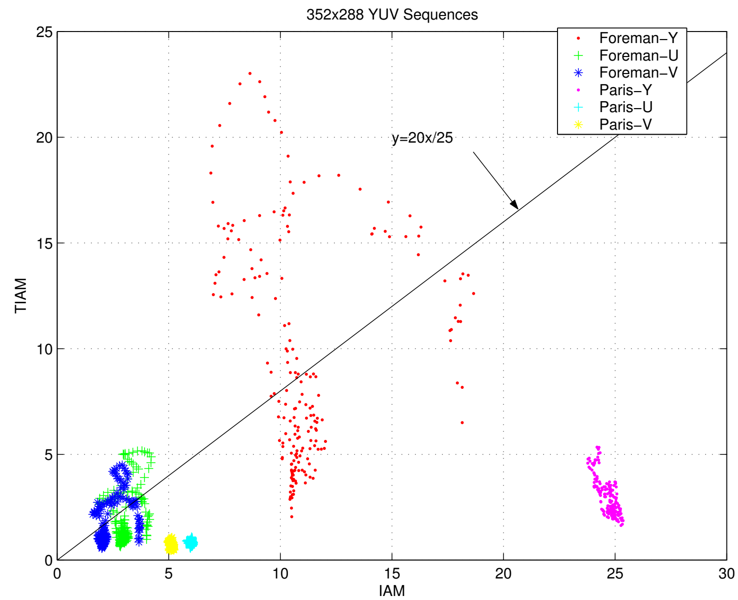


Fig. 4.14. Video Compressibility for the YUV 352×288 Foreman and Paris Sequences.

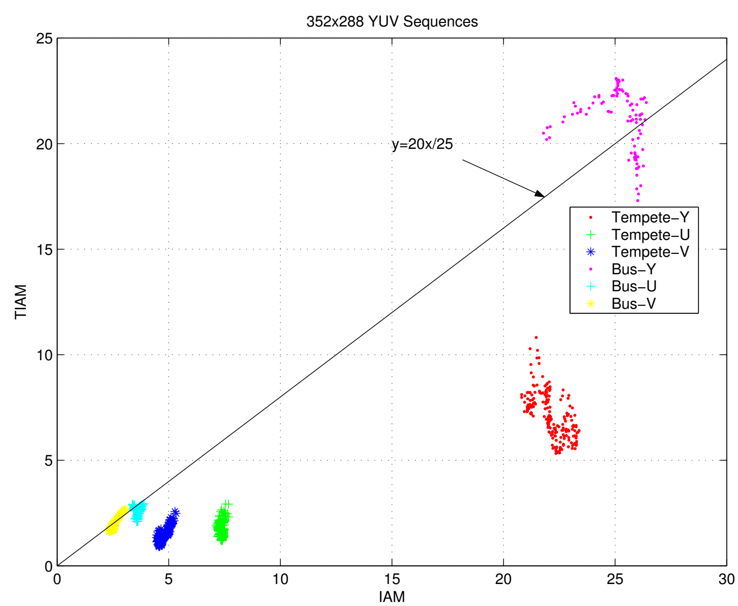


Fig. 4.15. Video Compressibility for the YUV 352×288 Tempete and Bus Sequences.

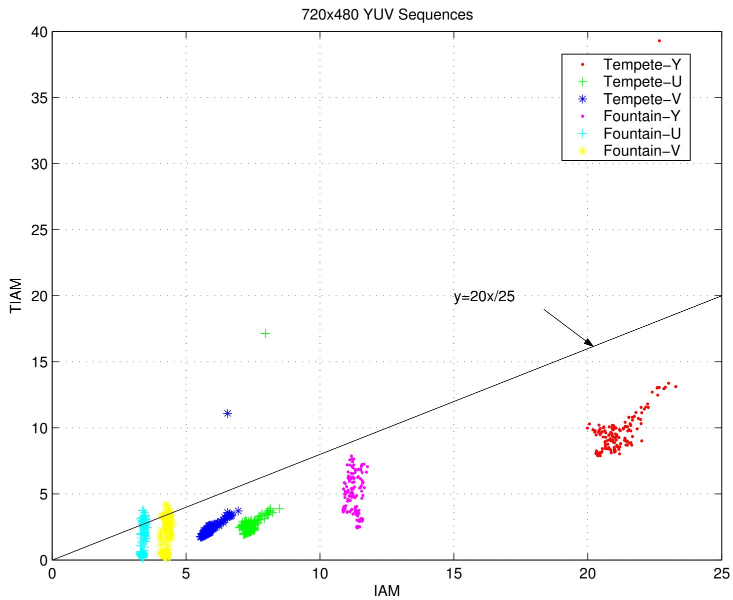


Fig. 4.16. Video Compressibility for the YUV 720×480 Tempete and Fountain Sequences.

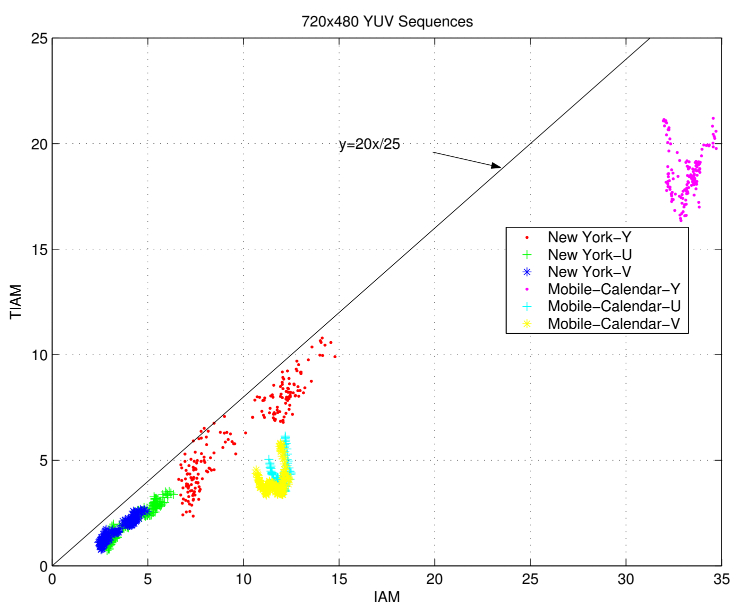


Fig. 4.17. Video Compressibility for the YUV 720×480 New York and Mobile-Calendar Sequences.

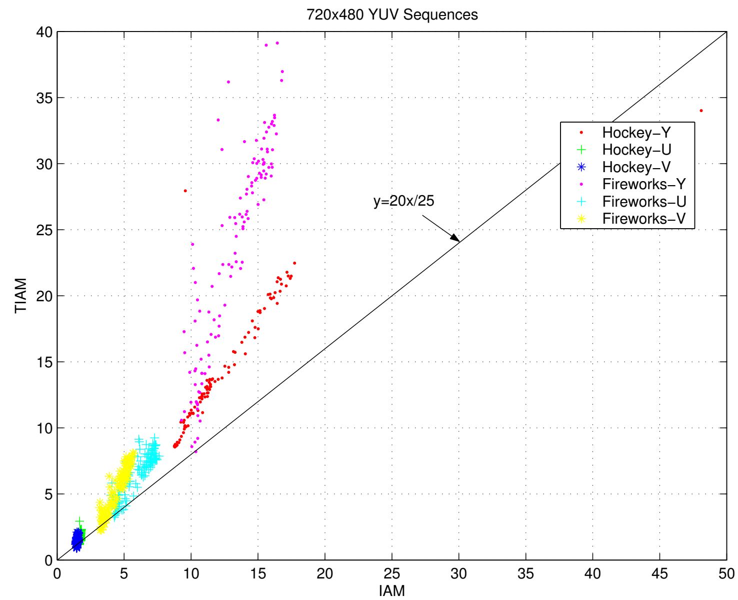


Fig. 4.18. Video Compressibility for the YUV 720×480 Hockey and Fireworks Sequences.

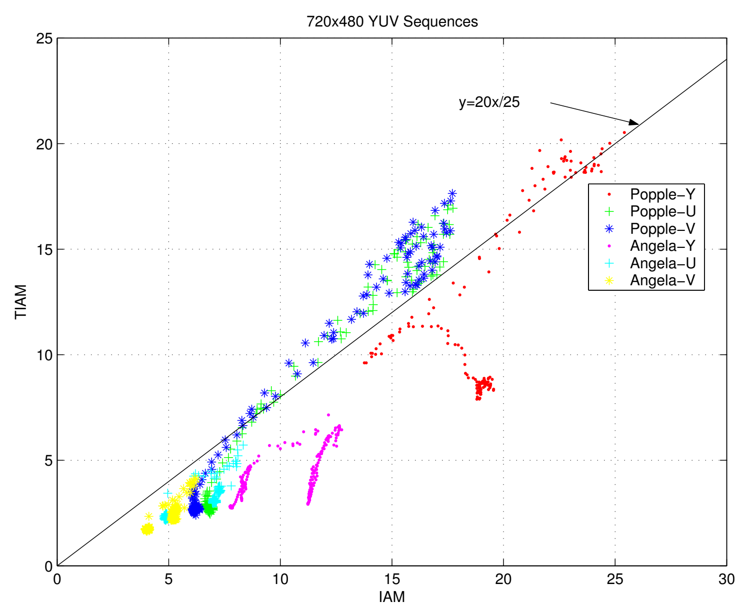


Fig. 4.19. Video Compressibility for the YUV 720×480 Popple and Angela Sequences.

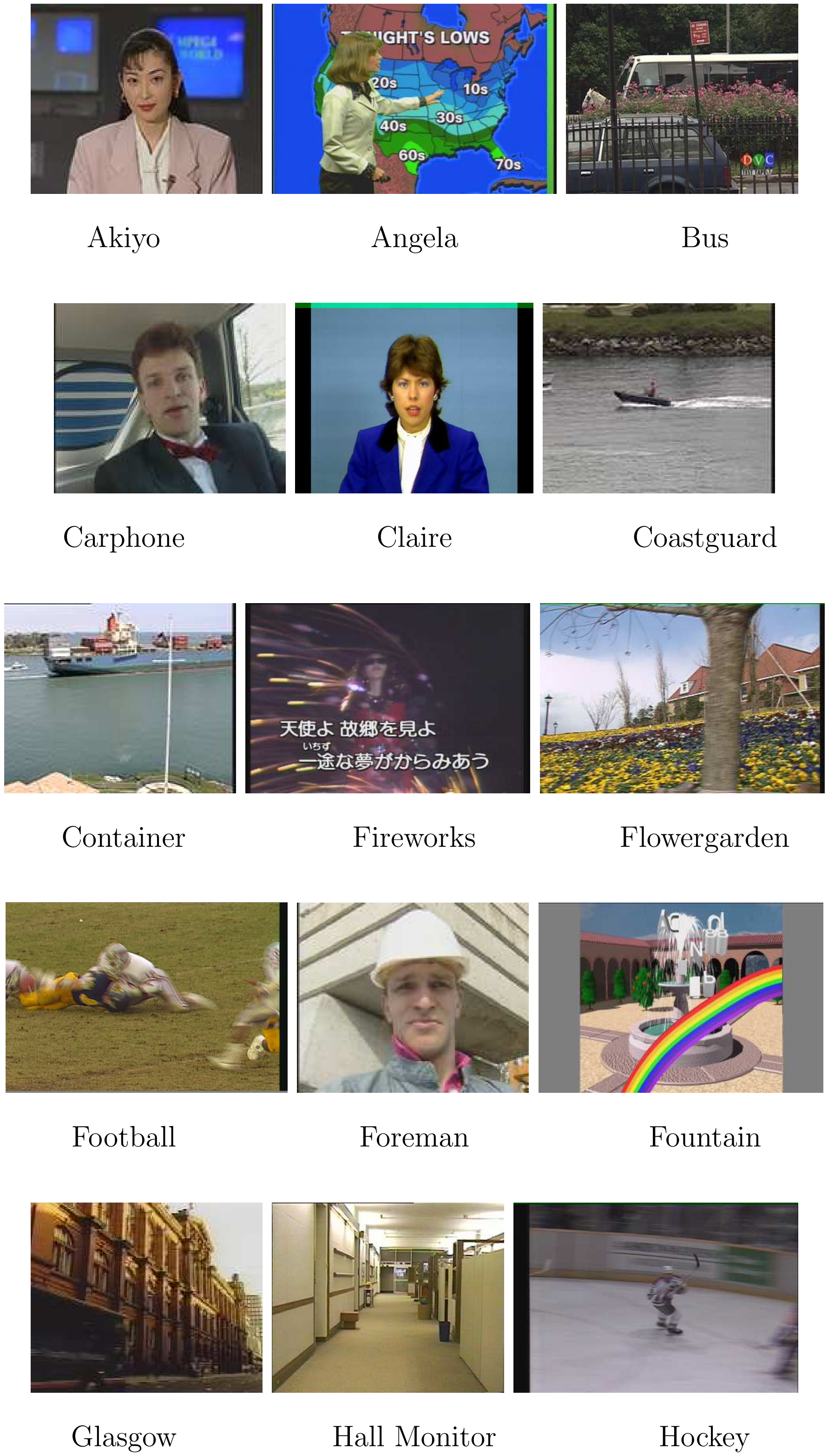


Fig. 4.20. Test Sequences 1.

Marbles Miss America MIT

Mobile-Calendar Mother-Daughter Music Player

News News1 News2

News2cut New York Paris

Popple Salesman Silent

Fig. 4.21. Test Sequences 2.

Tempete Tennis

Fig. 4.22. Test Sequences 3.

the test sequences are also encoded with the JPEG-LS [8] and the CALIC [7] lossless image compression techniques.

Special modes such as the “run mode” in JPEG-LS and the “binary mode” in CALIC designed for encoding flat regions were not used in the proposed video encoder since we are only interested in the compression performance contributed by the new backward-adaptive temporal prediction and the adaptive prediction mode. If the special mode for encoding flat regions is included in the encoder, further compression improvement can be obtained. Each pixel in the test sequence consists of 24 bits/pixel. Each frame is transformed into the RCT color space. Each color component is then treated independently. For the YUV test sequences, the format is 4:2:0 and no color transform is used.

4.9.1 Experiment 1: Compression Performance of the Proposed Algorithm

In the following tables and figures, *RCT* indicates the entropy of the reversible color transformed pixels for the RGB sequences and *ORG* represents the entropy of the original pixels for the YUV sequence. *JPEG-LS* and *CALIC* represent the data rate of the JPEG-LS and CALIC compression techniques. *MED* indicates the data rate of the compressed video sequence generated by the proposed algorithm using only spatial prediction. *ADAP* represents the data rate of the compressed video sequence using the proposed algorithm including adaptive prediction mode between spatial and temporal prediction. In the *ADAP* case, temporal prediction is used for macro blocks only if temporal prediction performed better than spatial prediction. Thus, the data rate reduction between *MED* and *ADAP* is obtained by incorporating temporal prediction. Table 4.6 shows the compression results of the RGB sequences with significant performance enhancement with respect to JPEG-LS and CALIC. For the experiments for JPEG-LS and CALIC, the RGB sequences are first converted by the reversible color transform. For the "Claire" sequence, the average data rate of *JPEG-LS* is 6.16 b/p, 6.68 b/p for *CALIC*, 6.47 b/p for *MED* and 5.55 b/p for *ADAP*. The average data rate reduction from *MED* to *ADAP* is 0.92 b/p, which is a 14.2 % reduction. This result includes the overhead of the prediction mode selection information, which must be transmitted to the decoder. The average data rate reduction from *JPEG-LS* to *ADAP* is 0.62 b/p, which is a 10.0 % reduction and the reduction from *CALIC* to *ADAP* is 1.14 b/p, which is a 17.0 % reduction.

Table 4.7 shows the compression results for the YUV sequences with a small frame size (176×144). The proposed algorithm shows excellent performance for these sequences. The average data rate reduction from *CALIC* to *ADAP* is 2.76 b/p, 2.56 b/p and 2.78 b/p , which are 64.1.0 %, 48.5 % and 54.4 % reductions for the "Akiyo", "Container" and "News" sequence respectively. This reduction comes from the high temporal correlation between successive frames due to small amounts of motion in these sequences. This result is expected from Figure 4.12 and 4.13 since the video compressibility for theses sequences is very high.

Tables 4.8 and 4.9 present compression results for video sequences with larger frame sizes. It is observed that the compression performance of the proposed algorithm is better than that of JPEG-LS and CALIC for most sequences. This result is also expected from Figure 4.14 and 4.15 since most points are below y.

Table 4.10 shows results for more video sequences. There is still a data rate reduction between *JPEG-LS* and *ADAP* for all the 720×480 test sequences. In the

Table 4.6
Data Rates (Bits/pixel) for the RGB Sequences.

Sequence		RCT	JPEG-LS	CALIC	MED	ADAP
360×288	Claire	16.695	6.163	6.683	6.466	**5.548**
	Miss America	18.501	8.243	9.520	8.626	**7.711**
720×486	Football	19.104	10.608	12.343	10.924	**9.193**
720×576	Mobile-Calendar	19.530	11.785	11.193	11.997	**10.749**
1600×960	MIT	20.428	9.496	10.998	9.843	**8.116**

performance comparison between *CALIC* and *ADAP*, some sequences have better results and some sequences do not. The "Fountain", "Mobile-Calendar", New York" and "Tempete" sequences have better result and this is expected from Figure 4.16 and 4.17. For the "Angela", "Fireworks", "Hockey" and "Popple" sequences, many points in Figure 4.18 and 4.19 are above y. Thus it is expected that the contribution of the temporal prediction will not be large. Instead, the use of better spatial prediction techniques such as GAP in CALIC contributes more to the compression performance shown in Table 4.10.

Figures 4.23, 4.24 and 4.25, show the data rate for the "Coastguard," "Mother-Daughter" and "Silent" 176×144 sequences. In Figure 4.23, the data rate of frames 79 and 80 are 6.1 b/p and 4.3 b/p respectively. The data rate of frame 80 is much smaller than that of frame 79. Frame 80 is very similar to the previous frame 79 (high temporal correlation), thus the prediction residuals of frame 80 have a smaller data rate due to the contribution of temporal prediction.

In Figures 4.26, 4.27 and 4.28, the data rates for the "Akiyo", "Bus" and "Hall Monitor" 352×288 sequences are shown. Figures 4.29, 4.30, 4.31, and 4.32 show the data rates for the "Flowergarden," "Football," "Angela" and "News2" 720×480 sequences.

Table 4.7
Data Rates (Bits/pixel) for the QCIF YUV Sequences.

Sequence		ORG	JPEG-LS	CALIC	MED	ADAP
176×144	Akiyo	9.580	4.330	4.310	4.277	**1.547**
	Carphone	9.524	4.975	4.934	4.955	**4.283**
	Claire	8.626	3.589	3.665	3.637	**2.533**
	Coastguard	8.919	6.018	5.989	5.988	**4.866**
	Container	9.073	5.262	5.281	5.177	**2.722**
	Foreman	9.434	5.869	5.679	5.845	**4.784**
	Glasgow	9.443	6.538	6.702	6.348	**5.608**
	Mother-Daughter	9.405	5.253	5.176	5.294	**3.813**
	News	9.354	5.063	5.108	4.945	**2.331**
	Salesman	9.057	5.864	5.779	5.705	**3.408**
	Silent	9.751	4.330	4.183	5.843	**3.488**

Table 4.8
Data Rates (Bits/pixel) for the CIF YUV Sequences.

Sequence		ORG	JPEG-LS	CALIC	MED	ADAP
352×288	Akiyo	9.622	3.999	3.929	4.058	**1.831**
	Bus	9.285	6.079	6.042	5.981	**5.676**
	Carphone	9.564	4.546	4.510	4.623	**4.407**
	Foreman	9.522	4.988	4.834	5.010	**4.703**
	Hall Monitor	9.411	4.904	4.814	4.896	**4.661**
	Paris	10.260	5.894	5.886	5.858	**3.567**
	Tempete	9.837	6.322	6.201	6.332	**5.709**

Table 4.9
Data Rates (Bits/pixel) for the YUV Sequences 1.

Sequence		ORG	JPEG-LS	CALIC	MED	ADAP
352×240	News2cut	9.065	4.195	4.161	4.287	**3.891**
	Tennis	9.208	7.305	7.145	7.411	**6.253**
360×240	News	9.085	4.193	4.162	4.283	**3.892**
360×288	Miss America	8.743	4.665	**4.595**	4.885	4.692
	Salesman	9.141	5.455	5.419	5.614	**4.699**
720×480	Angela	9.712	4.892	**4.739**	5.037	4.785
	Fireworks	9.515	5.254	**5.081**	5.371	5.213
	Flowergarden	10.650	7.128	6.930	7.292	**6.403**
	Football	9.603	6.511	6.298	6.567	**6.224**
	Fountain	8.212	3.574	3.501	3.786	**1.136**

Table 4.10
Data Rates (Bits/pixel) for the YUV Sequences 2.

Sequence		ORG	JPEG-LS	CALIC	MED	ADAP
720×480	Hockey	8.803	3.822	**3.642**	3.864	3.748
	Marbles	8.998	5.101	4.936	5.170	**4.845**
	Mobile-Calendar	10.822	7.499	7.366	7.691	**6.708**
	News1	9.623	4.674	4.576	4.762	**4.422**
	News2	9.309	4.285	**4.192**	4.362	4.215
	New York	9.476	4.695	4.540	4.816	**4.503**
	Popple	8.905	6.685	**6.338**	6.935	6.393
	Tempete	9.845	6.580	6.456	6.730	**6.388**
	Tennis	9.222	7.129	6.993	7.476	**6.567**
720x576	Music Player	9.458	6.257	6.073	6.500	**6.039**

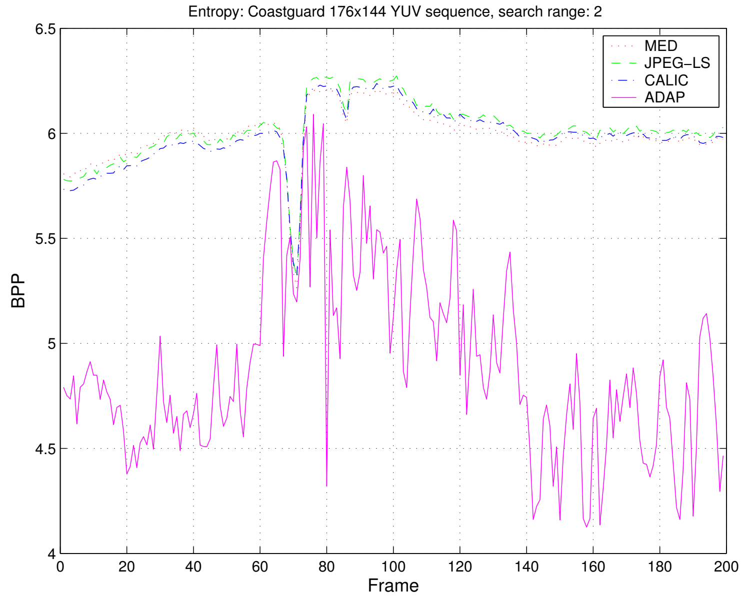


Fig. 4.23. Data Rates for the "Coastguard" YUV 176×144 Sequence with Search Range 2.

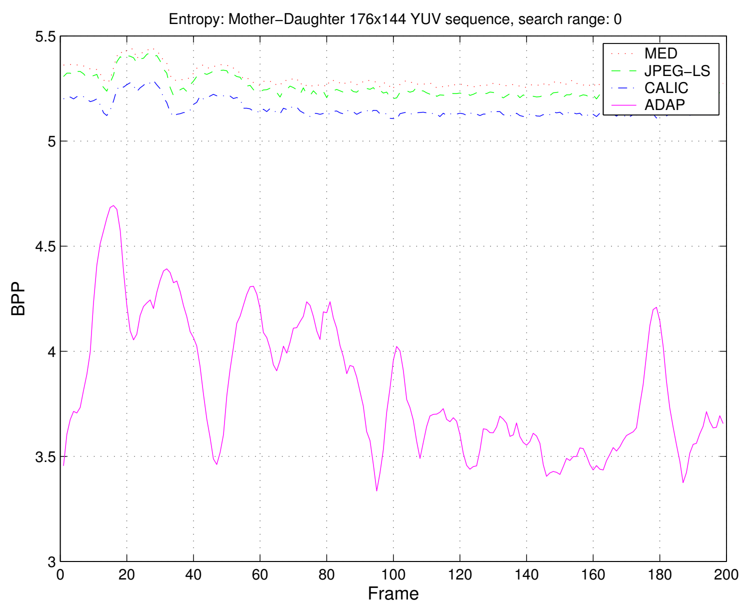


Fig. 4.24. Data Rates for the "Mother-Daughter" YUV 176×144 Sequence with Search Range 0.

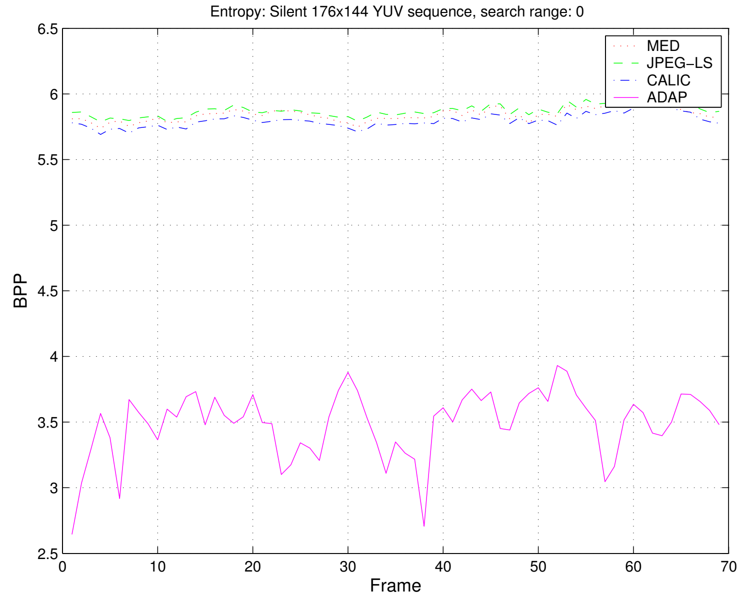


Fig. 4.25. Data Rates for the “Silent” YUV 176×144 Sequence with Search Range 0.

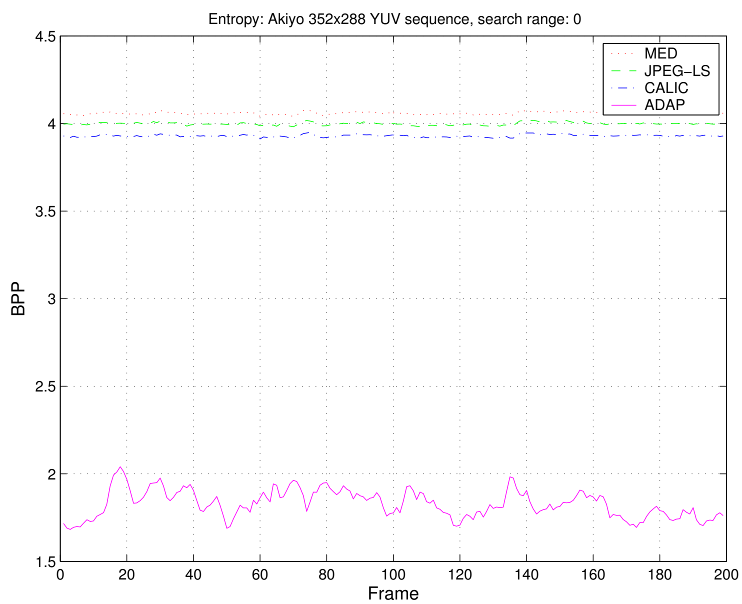


Fig. 4.26. Data Rates for the “Akiyo” YUV 352×288 Sequence with Search Range 0.

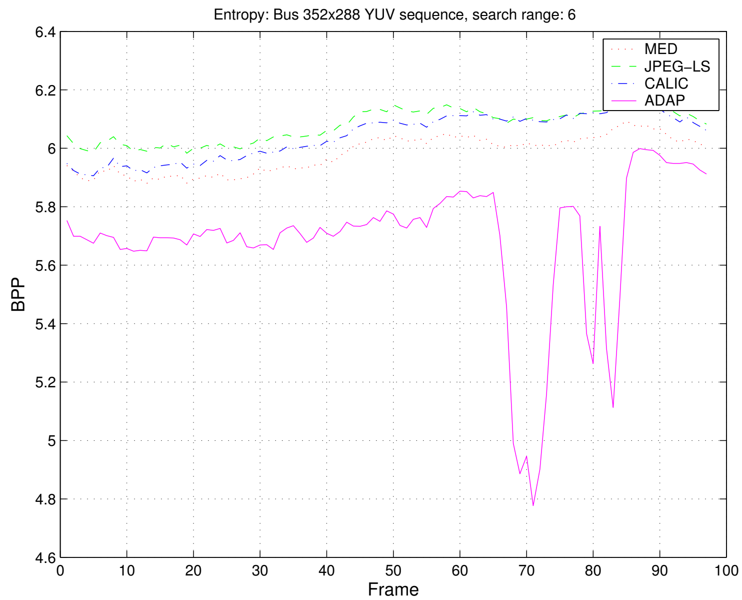


Fig. 4.27. Data Rates for the “Bus” YUV 352×288 Sequence with Search Range 6.

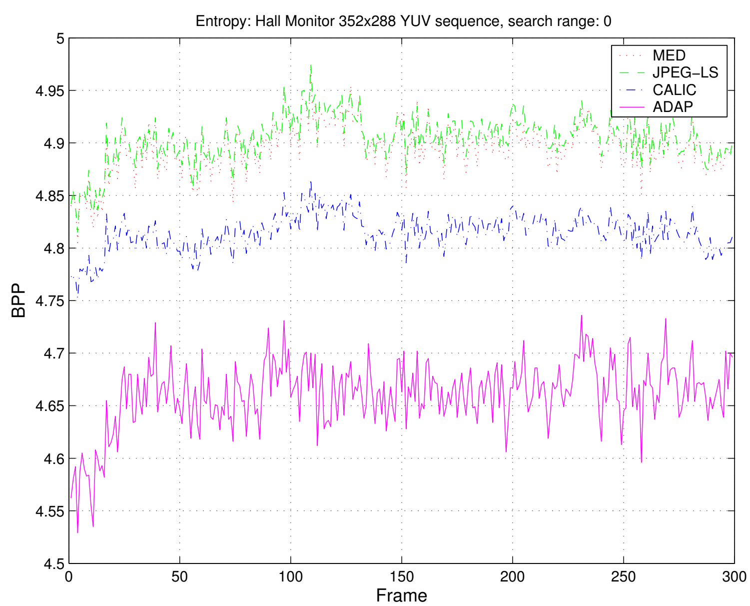


Fig. 4.28. Data Rates for the “Hall Monitor” YUV 352×288 Sequence with Search Range 0.

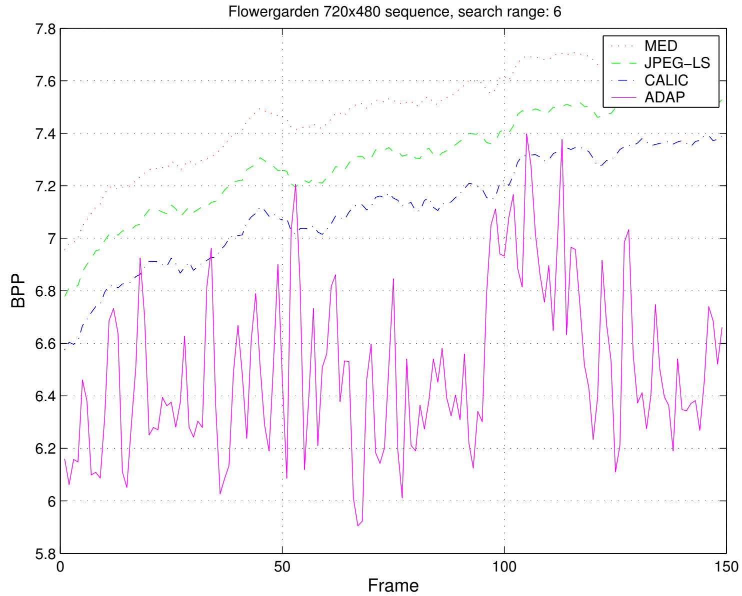


Fig. 4.29. Data Rates for the “Flowergarden” YUV 720×480 Sequence with Search Range 6.

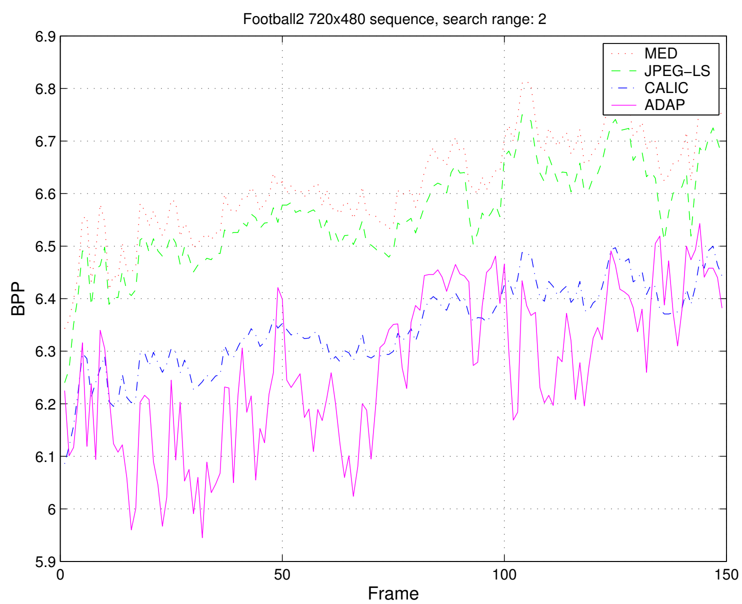


Fig. 4.30. Data Rates for the “Football” YUV 720×480 Sequence with Search Range 2.

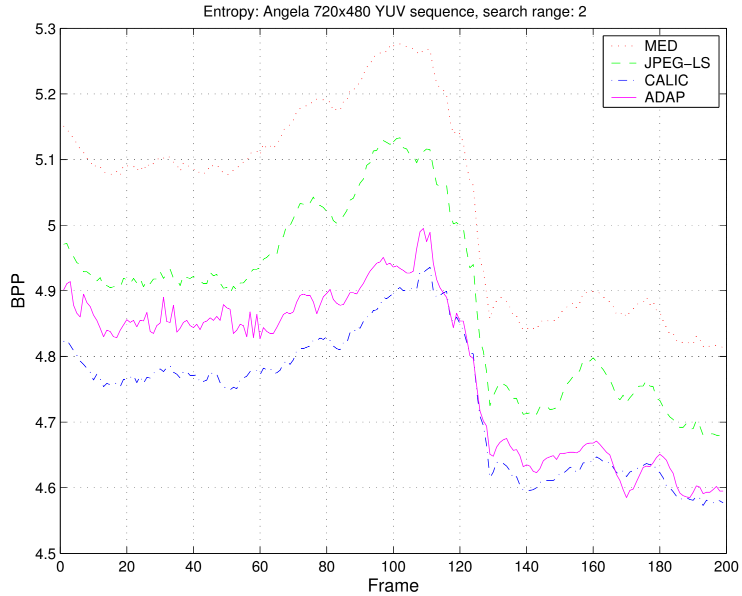


Fig. 4.31. Data Rates for the “Angela” YUV 720×480 Sequence with Search Range 2.

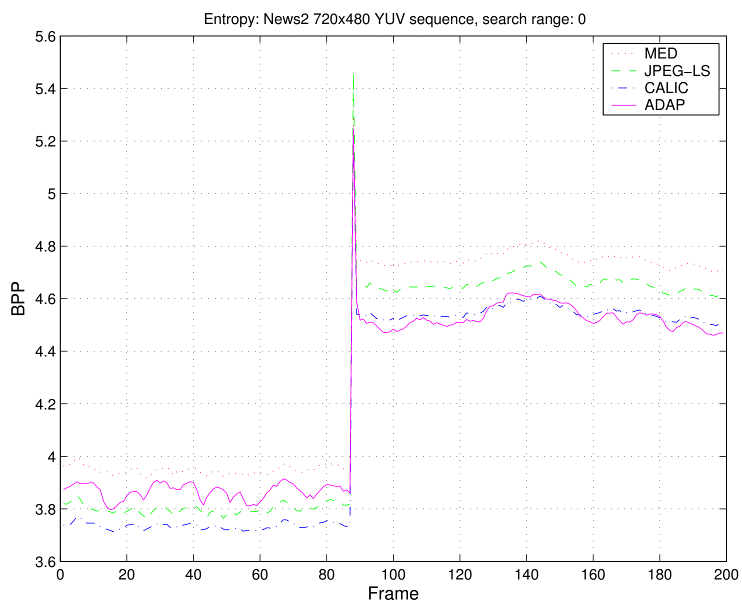


Fig. 4.32. Data Rates for the “News2” YUV 720×480 Sequence with Search Range 0.

4.9.2 Experiment 2: Video Compressibility and Frame Size

The average data rate reduction of the proposed algorithm becomes larger as the frame size becomes smaller as shown in Tables 4.7, 4.8, 4.9 and 4.10. A video sequence with a smaller frame size is obtained by subsampling of the original video sequence. A subsampling by two in the horizontal and vertical directions generates a $\frac{1}{4}$ frame size sequence. In this case, one pixel in the subsampled sequence represents four pixels in the original sequence. The distance between two adjacent pixels in the subsampled sequence is at least two in the horizontal and vertical directions in the original sequence. If the spatial correlation between pixels diminishes as the distance between the pixels becomes larger, the spatial correlation in the subsampled sequence may be smaller than that of the original sequence. That is, the IAM (Image Activity Measure) of the subsampled sequence is larger than that of the original sequence as shown in Table 4.11. The JPEG-LS utilizes only spatial correlation and we know that the compression performance of JPEG-LS is poorer on smaller frames as shown in Table 4.11. Thus, the compression performance of JPEG-LS depends on the IAM. The JPEG-LS compression performance of small IAM frames tends to be better than that of large IAM frames.

If a video sequence is subsampled by two, then the size of an object and motion is reduced in the subsampled video sequence as shown in Figure 4.33. In Figure 4.33, an object is out of the search region in the original sequence due to motion. In this case, motion estimation may not be good. However, the object is in the search region in the subsampled sequence due to the reduced object size and motion. In this case, motion estimation may work better. Thus, a video encoder utilizing motion estimation will work better for video sequences with a smaller frame size than for video sequences with a larger frame size. In the proposed new algorithm, temporal prediction by motion estimation is used when temporal prediction produces better result than spatial prediction for macro blocks. Thus, there are more opportunities to use temporal prediction for macro blocks when the frame size is small.

Table 4.11
Data Rates (Bits/pixel) Relative to the Frame Size.

Video Sequences		IAM	TIAM	Data Rates		
				ORG	JPEG-LS	ADAP
Akiyo	176×144	18.183	0.195	9.580	4.330	1.547
	352×288	12.744	1.176	9.622	3.999	1.831
Carphone	176×144	18.834	5.436	9.524	4.975	4.283
	352×288	13.845	6.297	9.564	4.546	4.407
Foreman	176×144	22.782	7.143	9.434	5.869	4.784
	352×288	16.515	13.200	9.522	4.988	4.703
Salesman	176×144	23.046	2.595	9.057	5.864	3.408
	360×288	19.545	5.028	9.141	5.455	4.692
Tempete	352×288	34.326	10.233	9.837	6.322	5.709
	720×480	34.473	14.700	9.845	6.580	6.388
Tennis	352×240	43.350	14.979	9.208	7.305	6.253
	720×480	36.378	14.853	9.222	7.129	6.567

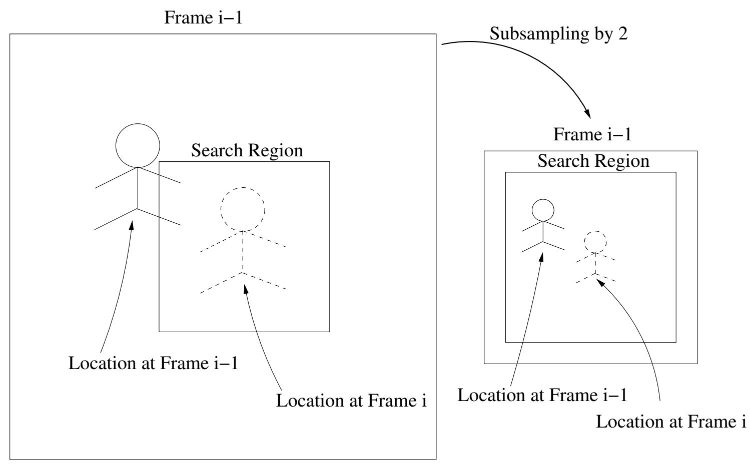


Fig. 4.33. Motion Estimation in Subsampled Video Sequence.

In Table 4.11, IAM of the “Akiyo” 176×144 sequence is 18.18 and is 12.74 for the “Akiyo” 352×288 sequence. The sequence with the larger frame size has a smaller IAM. The smaller IAM corresponds to higher image compressibility and this can be verified at the *JPEG-LS* column in Table 4.11. The JPEG-LS data rate of the “Akiyo” sequence with the larger frame size, 3.99 b/p, is smaller than that of the “Akiyo” sequence with the smaller frame size, 4.33 b/p. Now, the “Akiyo” 176×144 sequence has a smaller TIAM than the “Akiyo” 352×288 sequence. The smaller TIAM means higher video compressibility. This can be verified at the $ADAP$ column in Table 4.11. The proposed algorithm’s data rate ($ADAP$) of the “Akiyo” sequence with the smaller frame size, 1.55 b/p, is smaller than that of the “Akiyo” sequence with the larger frame size, 1.83 b/p. Similar results for other video sequences are also shown in Table 4.11.

It is concluded that the compression performance of the JPEG-LS becomes worse as the frame size becomes smaller since spatial correlation becomes also smaller. It is also concluded that the contribution of temporal prediction is significant for video sequences with a small frame size due to the improved motion estimation.

4.9.3 Experiment 3: Contribution of the Adaptive Selection Schemes

Three types of adaptation are used in the proposed algorithm. The first adaptation is done at the macro block (MB) level. Spatial and temporal prediction techniques are used on each macro block and the SADs (Sum of Absolute Difference) of each technique are obtained. The prediction technique generating the smaller SAD is selected as the prediction mode of the macro block. The next adaptation is obtained at the subband level. If the IWT is not used, then the entire frame will be a subband. For each subband, the entropy is obtained for the prediction residuals obtained by using spatial and/or temporal prediction at the macro block level (*Prediction* mode). The entropy of the original subband is also obtained (*Direct Sending* mode). The entropies of the two methods are compared and the prediction mode with the smaller entropy is selected as the prediction mode for the subband (*Prediction* mode or *Direct Sending* mode). Another adaptation is used at the frame level. The compression result using the IWT is compared with the compression result without using the IWT and the better technique is selected for the frame.

In this section, the contribution of each adaptation mode will be investigated. Tables 4.12 and 4.13 show the compression results in bits/pixel for each adaptation technique. If the value of the adaptation at the frame level is 0, this means that the IWT is not used. If the value of the adaptation at the frame level is 1, the IWT is used. If the value of the adaptation at the subband level is 0, then the subband is in the *Prediction* mode and spatial and/or temporal prediction are always used on the subband. If the value of the adaptation at the subband level is 1, then the adaptive selection between the *Prediction* and *Direct Sending* mode is done for the subband. If the value of the adaptation at the macro block level is 0, only spatial prediction is used for the macro block. If the value of the adaptation at the macro block level is 1, the adaptive prediction mode between spatial and temporal prediction is used for the macro block. The data rate reduction caused by this adaptation is due to temporal prediction technique.

Table 4.12
Data Rates (Bits/pixel) relative to the Adaptive Selection of Prediction Mode for Video Sequences 1.

IWT	Adaptations		Video Sequences				
	Subband Level	MB Level	Angela	Firework	Fountain	Marbles	Music Player
0	0	0	5.037	5.371	3.786	5.170	6.500
	0	1	4.898	5.221	1.136	4.987	6.038
	1	0	5.037	5.371	3.786	5.170	6.500
	1	1	4.898	5.221	**1.136**	4.987	**6.039**
1	0	0	5.257	5.598	2.070	5.115	6.384
	0	1	5.004	5.390	1.380	4.944	6.292
	1	0	5.023	5.272	4.486	4.954	6.301
	1	1	**4.785**	**5.213**	1.380	**4.845**	6.086

Table 4.13
Data Rates (Bits/pixel) relative to the Adaptive Selection of Prediction Mode for Video Sequences 2.

IWT	Adaptations		Video Sequences				
	Subband Level	MB Level	News1	News2	New York	Popple	Tempete
0	0	0	4.762	4.362	4.816	6.935	6.730
	0	1	4.468	4.248	4.503	6.597	6.438
	1	0	4.762	4.362	4.816	6.597	6.731
	1	1	4.468	4.248	**4.503**	6.597	6.438
1	0	0	4.851	4.535	4.750	6.758	6.917
	0	1	4.624	4.490	4.735	6.528	6.579
	1	0	4.805	4.394	4.788	6.488	6.669
	1	1	**4.422**	**4.215**	4.652	**6.393**	**6.388**

The adaptation modes generating the best compression results are selected and the resulted data rates are represented as the bold numbers in Table 4.12 and 4.13. The triplet (0,0,0) represents the adaptation modes for IWT, subband level and macro block level. (0,0,0) indicates that the IWT is not used, the adaptation at the subband level is not used (*Prediction* mode) and the adaptation at the macro block level is not used either (only spatial prediction). For "Fountain" in Table 4.12, the data rate is 3.786 b/p in the (0,0,0) mode and is 1.136 b/p in the (0,0,1) mode. The data reduction 2.65 b/p is obtained due to the adaptive prediction mode at the macro block level, mainly the contribution of temporal prediction. The data rate is 1.136 b/p for the (0,1,1) mode, which is the same as the (0,0,1) mode and the data rate is 3.786 b/p for the (0,0,0) mode, which is the same as the (0,1,0) mode. These results show that the adaptation at the subband level does not contribute to the compression performance if the IWT is not used.

For "Angela" in Table 4.12, the data rate is 5.257 b/p for the (1,0,0) mode and is 5.004 b/p for the (1,0,1) mode. The data reduction 0.253 b/p due to the adaptation at the macro block level is significant. The data rate reduction 0.219 b/p obtained by the adaptation at the subband level is also significant since the data rate is reduced from 5.004 b/p for the (1,0,1) mode to 4.785 b/p for the (1,1,1) mode. The data rate is 5.023 b/p for the (1,1,0) mode, which is also significant data rate reduction from the (1,0,0) mode due to the adaptation at the subband level. The data rate is also reduced significantly from the (1,1,0) mode to the (1,1,1) mode. Thus both adaptations at the subband level and at the macro block level have significant contribution to the compression performance.

When the IWT is not used, the adaptive prediction mode at the macro block level contributes most to the compression performance. Thus the new backward-adaptive temporal prediction is the most important. When the IWT is used, the adaptation at the subband level is also important in addition to the adaptation at the macro block level.

5. CONCLUSIONS AND FUTURE WORK

In this thesis, a new lossless compression algorithm for color video sequences has been presented. A new temporal prediction technique was also developed. The new temporal prediction technique is similar to the concept of the use of motion vectors, but requires no side information. The adaptive selection of prediction mode between temporal and spatial prediction was also investigated. A quantitative method to adaptively determine which IWT should be used for a given frame in a video sequence was shown. The decision was made by examining the uniformity and the image activity measure (IAM). The performance comparisons with respect to the search range in temporal prediction were also described. Using the new backward-adaptive temporal prediction and the adaptive selection method between spatial and temporal prediction the new scheme was shown to be better than the state-of-the-art lossless compression algorithms.

For future work, the proposed algorithm is based on the spatial prediction technique used in JPEG-LS and always has better compression performance than JPEG-LS due to the new prediction techniques. Typically CALIC shows slightly better compression performance than JPEG-LS with higher complexity. If the new prediction techniques such as the backward-adaptive temporal prediction and the adaptive prediction mode are used with a CALIC based algorithm, then the performance of the new algorithm will be better than that of the CALIC algorithm.

There are two methods for improving the compression performance of a video encoder. One is to produce prediction residuals with smaller entropy. The other is to encode the prediction residuals more efficiently. A smaller entropy of the prediction residuals can be obtained by more accurate prediction of the original frames. The search range for motion estimation is fixed for a video sequence. In order to improve the performance of motion estimation, the best search range will be obtained for each block. The search range information for each block should be transmitted to the decoder. The best search range may be the same for several blocks in static areas of a frame. In this case, sending one search range for several blocks will save side information. A quad-tree can be used for encoding the search range.

In the proposed new algorithm, a simple context modeling is used for entropy coding. Better compression can be achieved by using the more complex context modeling used in JPEG-LS and CALIC.

The proposed new algorithm is designed to encode video sequences losslessly. A lossy version of the proposed algorithm can be implemented by incorporating a quantization step. The quantization is used on the prediction residual frames in order to reduce the entropy. The quantization step size can be adaptively modified in order to produce a bitstream with a given data rate budget.

LIST OF REFERENCES

LIST OF REFERENCES

[1] JPEG: ISO/IEC JTC1/SC29 WG10, *Information Technology - Digital Compression and Coding of Continuous- tone Still Images - Requirement and Guidelines*, 1993. Final Draft International Standard ISO/IEC 10918-1.

[2] ITU-T, *ITU-T Recommendation T.800: CD15444-1, V1.0*, December 2000. (JPEG2000).

[3] ITU-T, *ITU-T Recommendation H.263: Video Coding for Low Bitrate Communication*, March 1996.

[4] MPEG-4: ISO/IEC JTC1/SC29/WG11 N2502, *ISO/IEC 14496-2, Coding of Audio-visual Objects: Visual*, October 1998. (MPEG-4 Video 1).

[5] J. Korris and M. Macedonia, "The end of celluloid: Digital cinema emerges," *Computer*, vol. 35, no. 4, pp. 96–98, April 2002.

[6] ISO/IEC JTC1/SC29 WG11 N5328, *Digital Cinema Requirements*, December 2002. Coding of Moving Pictures and Audio.

[7] X. Wu and N. D. Memon, "Context-based, adaptive, lossless image coding," *IEEE Transactions on Communication*, vol. 45, no. 4, pp. 437–444, April 1997.

[8] JPEG-LS, *information Technology - Lossless and Near-lossless Compression of Continuous-tone Still Images*, 1998. Final Draft International Standard FDIS14495-1.

[9] A. J. Penrose and N. A. Dodgson, "Extending lossless image compression," *Eurographics UK '99*, April 1999.

[10] E. S. G. Carotti, J. C. D. Martin, and A. R. Meo, "Backward-adaptive lossless compression of video sequences," *Proceedings of the IEEE International Conference on Audio, Speech and Sinal Processing*, vol. 4, pp. 3417–3420, May 2002.

[11] N. D. Memon and K. Sayood, "Lossless compression of video sequences," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1340–1345, Oct 1996.

[12] B. B. Chai, J. Vass, and X. Zhuang, "Significance-linked connected component analysis for wavelet image coding," *IEEE Transactions on Image Processing*, vol. 8, no. 6, pp. 774–784, June 1999.

[13] F. Sheng, A. Bilgin, P. J. Sementilli, and M. W. Marcellin, "Lossy and lossless image compression using reversible integer wavelet transforms," *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, pp. 876–880, October 1998.

[14] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet transforms that map integers to integers," *Applied and Computational Harmonic Analysis*, vol. 5, no. 3, pp. 332–369, July 1998.

[15] M. Grangetto, E. Magli, M. Martina, and G. Olmo, "Optimization and implementation of the integer wavelet transform for image coding," *IEEE Transactions on Image Processing*, vol. 11, no. 6, pp. 596–604, June 2002.

[16] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Lossless image compression using integer to integer wavelet transforms," *IEEE International Conference on Image Processing*, vol. 1, pp. 596–599, October 1997.

[17] W. Sweldens, "The lifting scheme: A construction of second generation wavelets," *SIAM Journal on Mathematical Analysis*, vol. 29, no. 2, pp. 511–546, 1998.

[18] R. L. Claypoole, R. G. Baraniuk, and R. D. Nowak, "Lifting construction of non-linear wavelet transforms," *International Symposium on Time-Frequency and Time-Scale Analysis*, pp. 49–52, October 1998.

[19] M. D. Adams and A. Antoniou, "Reversible ezw-based image compression using best-transform selection and selective partial embedding," *IEEE Transactions on Circuits and Systems*, vol. 47, no. 10, pp. 1119–1122, October 2000.

[20] S. Saha and R. Vemuri, "An analysis on the effect of image activity on lossy coding performance," *Proceedings of International Symposium on Circuits and Systems*, vol. III, pp. 295–298, Geneva, Switzerland, May 2000.

[21] K. R. Rao and P. Yip, *Discrete Cosine Transform - Algorithms, Advantages, Applications*. New York: Academic Press, 1990.

[22] S. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, July 1989.

[23] M. Antonini, M. Barlaud, and P. Mathieu, "Image coding using wavelet transform," *IEEE Transactions on Image Processing*, vol. 1, no. 2, pp. 205–220, April 1992.

[24] M. Vetterli and C. Herley, "Wavelets and filter banks: Theory and design," *IEEE Transactions on Acoustics and Speech Signal Processing*, vol. 40, no. 9, pp. 2207–2232, September 1992.

[25] S. A. Martucci, I. Sodagar, T. Chiang, and Y. Zhang, "A zerotree wavelet video coder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, pp. 109–118, February 1997.

[26] MPEG-2: ISO/IEC, *ISO/IEC 13818-2:2000, Information Technology - Generic coding of moving pictures and associated audio information: Video*, 2000. (MPEG-2 Video).

[27] J. M. Shapiro, "Embedded image coding using zerotree of wavelet coefficinets," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, December 1993.

[28] A. Said and W. A. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, June 1996.

[29] A. K. Jain, *Fundamentals of Digital Image Processing*. Englewood Cliffs NJ: Prentice Hall, 1989.

[30] S. W. Golomb, "Run-length encodings," *IEEE Transactions on Information Theory*, vol. IT-12, pp. 399–401, July 1966.

[31] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, September 1952.

[32] J. Rissanen, "Generalized kraft inequality and arithmetic coding," *IBM Journal on Research and Development*, vol. 20, no. 3, pp. 198–203, May 1976.

[33] J. Rissanen and J. G. G. Langdon, "Generalized kraft inequality and arithmetic coding," *IBM Journal on Research and Development*, vol. 23, no. 2, pp. 149–162, March 1979.

[34] P. G. Howard and J. S. Vitter, "Fast and efficient lossless image compression," *Proceedings of IEEE Data Compression Conference*, pp. 351–360, Snowbird, Utah, March 1993.

[35] A. Bilgin, P. J. Sementilli, F. Sheng, and M. W. Marcellin, "Scalable image coding using reversible integer wavelet transforms," *IEEE Transactions on Image Processing*, vol. 9, no. 11, pp. 1972 – 1977, November 2000.

[36] A. Zandi, J. D. Allen, E. L. Schwartz, and M. Boliek, "CREW: Compression by reversible embedded wavelets," *Proceedings of IEEE Data Compression Conference*, pp. 212–221, March 1995.

[37] I. Daubechies and W. Sweldens, "Factoring wavelet transform into lifting steps," *Journal of Fourier Analysis and Applications*, vol. 4, no. 3, pp. 247–269, 1998.

[38] I. H. Witten, R. Neal, and J. M. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, no. 6, pp. 520–540, June 1987.

[39] J. J. Hwang, M. H. Lee, and K. R. Rao, "Transmission noise eliminations in BDPCM image," *IEEE Transactions on Consumer Electronics*, vol. 39, no. 3, pp. 151–158, August 1993.

[40] V. R. Algazi and D. J. Sakrison, "On the optimality of karhunen-loeve expansion," *IEEE Transactions on Information Theory*, vol. IT-15, no. 3, pp. 319–321, March 1969.

[41] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform," *IEEE Transactions on Signal Processing*, vol. 40, no. 9, pp. 2174–2193, September 1992.

[42] N. Ahmed, T. Natarajan, and K. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. 23, no. 23, pp. 90–93, January 1974.

[43] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG Video Compression Standard*. New York: Chapman and Hall, 1996.

[44] V. K. Heer and H.-E. Reinfelder, "A comparison of reversible methods for data compression," *Proceedings of SPIE Medical Imaging*, vol. 1233, pp. 354–365, 1990.

[45] A. Said and W. A. Pearlman, "An image multiresolution representation for lossless and lossy image compression," *IEEE Transactions on Image Processing*, vol. 5, pp. 1303–1310, September 1996.

[46] F. Dufaux and F. Moscheni, "Motion estimation techniques for digital TV: A review and a new contribution," *Proceedings of the IEEE*, vol. 83, no. 6, pp. 858–876, June 1995.

[47] K. M. Nam, J.-S. Kim, R.-H. Park, and Y. S. Shim, "A fast hierarchical motion vector estimation algorithm using mean pyramid," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 4, pp. 344–351, August 1995.

[48] M. Bierling, "Displacement estimation by hierarchical blockmatching," *Proceedings of SPIE Conference on Visual Communication and Image Processing*, vol. 1001, pp. 942–951, Cambridge, MA, November 1988.

[49] ITU-T Rec. H.264/ISO/IEC 11496-10, *Advanced Video Coding*, March 2003. Final Committee Draft.

[50] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264avc video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, July 2003.

[51] M. Wien, "Variable block-size transforms for h.264avc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 604–613, July 2003.

[52] C. E. Shannon, "A mathematical theory of communications: Parts I and II," *Bell System Technology Journal*, vol. 27, pp. 379–423, 623–656, July and October 1948.

[53] T. Cover and J. Thomas, *Elements of Information Theory.* New York: John Wiley & Sons, Inc, 1991.

[54] A. Papoulis, *Probability, Random Variables, and Stochastic Processes.* Boston: WCB/McGraw-Hill, third ed., 1991.

[55] A. M. Moffat, R. M. Neal, and I. H. Witten, "Arithmetic coding revisited," *IEEE Transactions on Information Systems*, vol. 16, no. 3, pp. 256–294, July 1998.

[56] M. D. Adams and F. Kossentini, "Performance evaluation of reversible integer-to-integer wavelet transforms for image compression," *Proceedings of IEEE Data Compression Conference*, pp. 514–523, March 1999.

[57] D. Marpe, G. Blattermann, J. Ricke, and P. Maab, "Two-layered wavelet-based algorithm for efficient lossless and lossy image compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 7, pp. 1094–1102, October 2000.

[58] V. N. Ramaswamy, K. R. Namuduri, and N. Ranganathan, "Context-based lossless image coding using EZW framework," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 1–6, April 2001.

[59] K.-W. Cheung and L.-M. Po, "Spatial coefficient partitioning for lossless wavelet image coding," *IEE Proceedings of Vision, Image and Signal Processing*, pp. 365–369, December 2002.

[60] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image Compression fundamentals, standards and Practice.* Norwell MA: Kluwer Academic Publishers, 2002.

[61] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Transactions on Image Processing*, vol. 9, no. 7, pp. 1158–1170, July 2000.

[62] D. Santa-Cruz and T. Ebrahimi, "A study of JPEG-2000 still image coding versus other standards," *Proceedings of the X European Signal Processing Conference*, vol. 2, pp. 673–676, Tampere, Finland, September 2000.

[63] G. K. Wallace, "Digital image and video standards," *Communication of the ACM*, vol. 34, no. 4, pp. 31–44, April 1991.

[64] B. Carpentieri, M. J. Weinberger, and G. Seroussi, "Lossless compression of continuous-tone images," *Proceedings of the IEEE*, vol. 88, no. 11, pp. 1797–1809, November 2000.

[65] M. J. Weinberger, G. Seroussi, and G. Sapiro, "LOCO-I: A low complexity, context-based, lossless image compression algorithm," *Proceedings of IEEE Data Compression Conference*, pp. 140–149, Snowbird, Utah, March 1996.

[66] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309–1324, August 2000.

[67] S. D. Rane and G. Sapiro, "Evaluation of JPEG-LS, the new lossless and controlled-lossy still image compression standard, for compression of high-resolution elevation data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 39, no. 10, pp. 2298–2306, October 2001.

[68] X. Wu and N. Memon, "Context-based lossless interband compression - extending CALIC," *IEEE Transactions on Image Processing*, vol. 9, no. 6, pp. 994–1001, June 2000.

[69] M. Domanski and K. Rakowski, "Lossless and near-lossless image compression with color transformations," *Proceedings of the IEEE International Conference on Image Processing*, vol. 3, pp. 454–457, 2001.

[70] L. Goffman-Vinopal and M. Porat, "Color image compression using inter-color correlation," *Proceedings of the IEEE International Conference on Image Processing*, vol. 2, pp. 353–356, 2002.

[71] J. Jiang, "Variations of JPEG-LS and its applications." http://citeseer.nj.nec.com/jiang00variations.html.

[72] E. A. Edirisinghe, S. Bedi, and C. Grecos, "Improvements to JPEG-LS via diagonal edge based prediction," *Proceedings of SPIE Visual Communication and Image Processing*, vol. 4671, no. 2, pp. 604–613, January 2002.

[73] R. Barequet and M. Feder, "SICLIC: A simple inter-color lossless image coder," *Proceedings of the IEEE Data Compression Conference*, pp. 501–510, March 1999.

[74] S. V. Assche, W. Philips, and I. Lemahieu, "Lossless compression of pre-press images using a novel color decorrelation technique," *Proceedings of the International Conference on Image Analysis and Processing*, vol. 1311, pp. 85–92, Florence Italy, September 1997.

[75] D. Brunello, G. Calvagno, G. A. Mian, and R. Rinaldo, "Lossless video coding using optimal 3D prediction," *Proceedings of the IEEE International Conference on Image Processing*, pp. 89–92, September 2002.

[76] D. Brunello, G. Calvagno, G. A. Mian, and R. Rinaldo, "Lossless compression of video using temporal information," *IEEE Transactions on Image Processing*, vol. 12, no. 2, pp. 132–139, February 2003.

VITA

VITA

Sahng-Gyu Park received his B.S. degree in electrical engineering from Kyungpook National University, Taegu, Korea, in 1989 and his M.S degree in the school of Electrical and Electronics Engineering from Korea Advanced Institute of Science and Technology (KAIST), Seoul, Korea, in 1991. From 1991 to 1998, he was with Korea Telecom, where he was involved in the development of speech recognition systems for stock information and voice dialing. In 1998, he started his studies toward the Ph.D. degree in the School of Electrical and Computer Engineering at Purdue University, West Lafayette, Indiana. From 2000 to 2003, Park was a teaching assistant at Purde University. He joined the Video and Image Processing Laboratory in 2000. In the summer of 2002, he worked as a summer intern for the Corporate Research Group at Thomson. His research interests are in the areas of video and image coding, MPEG-2, MPEG-4, H.263, H.264, JPEG-2000, JPEG-LS, lossless compression and scalable video compression.