# CODING OF MOTION PICTURE RESIDUES WITHOUT SPATIAL TRANSFORMS

YAN SHU

# CODING OF MOTION PICTURE RESIDUES WITHOUT SPATIAL TRANSFORMS

BY

©Yan Shu. B. ENG.

JANUARY 1999

A THESIS SUBMITTED TO THE SCHOOL OF GRADUATE

STUDIES IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING

FACULTY OF ENGINEERING AND APPLIED SCIENCE

MEMORIAL UNIVERSITY OF NEWFOUNDLAND

ST. JOHN'S, NEWFOUNDLAND

St. John's Newfoundland Canada

# Abstract

A coding scheme for prediction-image residues in video coding is developed in this thesis. The strategy is to use the zerotree structure invented by Shapiro and extended by Said and Pearlman. Spatial patterns of pixels are defined to occupy successive levels of the tree structure. These patterns replace the standard quad-tree ordering of wavelet coefficient values with simple pixel values. The motivation for this is that prediction image residues have little spatial structure. Simulations to evaluate the strategy are included in the thesis. The results demonstrate that video coding with this simple zerotree pattern coding algorithm for motion picture-residues without using any spatial transforms can achieve good trade-offs between compression performance and channel error resilience and robustness, and provide additional error concealment with the help of a simple post-processing technique. It allows for operation in noisy channel conditions, without the aid of error correction or detection techniques.

# Acknowledgements

My deepest thanks go to my supervisor Dr. John A. Robinson, who through his interest and concern for multimedia communication, image processing, and graduate studies has brought more into my scientific life than any other person. His friendliness, advice and support encouraged me throughout my studying. I am proud to have had him as my supervisor.

I am particularly grateful to my colleague Li-Te Cheng with whom I had the opportunity to discuss ideas. His comments and valuable suggestions have been of great help during the two years.

I also express a special thank you to all the members of the Multimedia Communications Lab for the discussions regarding my research and the nice and happy working environment we create together.

I especially thank my parents, my sister and brother in China for their love, encouragement, and support.

Finally, financial support from Dr. John A. Robinson, the School of Graduate Studies and the Faculty of Engineering and Applied Science at Memorial University of Newfoundland is grateful acknowledged.

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations and Acronyms

| | |
|---|---|
| BER | Bit Error Rate |
| CIF | Common Intermediate Format |
| CRC | Cyclic Redundancy Codes |
| DCT | Discrete Cosine Transform |
| DWT | Discrete Wavelet Transform |
| ECC | Error Control Coding |
| EREC | Error Resilient Entropy Code |
| ERPC | Error Resilient Positional Code |
| EZW | Embeded Zerotree Wavelet algorithm |
| IEC | International Electrotechnical Commission |
| ISO | International Standardisation Organization |
| ITU | International Telecommunication Union |
| JPEG | Joint Photographic Experts' Group |
| MAE | Mean Absolute Error |
| MPEG | Moving Picture Experts' Group |
| MSE | Mean Square Error |
| PSNR | Peak Signal to Noise Ratio |
| QCIF | Quarter CIF |
| SIF | Source Intermediate Format |
| SPIHT | Set Partitioning In Hierarchical Tree algorithm |
| VLC | Variable-Length Coding |

# Chapter 1

# Introduction

Modern telecommunication services for new applications such as video-telephony, wireless image transmission, electronic newspapers, interactive multimedia databases, and so on require transmission at low bit rates implying a high degree of image sequence compression. The principle of compression algorithms is to reduce the redundancy existing in video signals, which is of two types: spatial (within a single frame) and temporal (between adjacent frames). Over the past twenty years, many video compression techniques and standards have been developed and implemented, such as Intel RTV/Indeo, Intel PLV, IBM Photomotion, Motion JPEG, Fractals, Wavelets, H.261/H.263, and MPEG.

Generally, video compression algorithms include a two phase process. In the first phase, motion estimation and motion compensation techniques are applied for the reduction of temporal redundancy between video frames. With motion estimation, the displacement of objects in the picture is estimated, and then this motion information is used to form a prediction from already-transmitted pictures. In the second phase, a motion-compensated difference frame (residue) is constructed by taking pointwise differences between the picture element values in the input frame and the corresponding prediction frame. The motion vectors and difference frames are transmitted.

In most video compression techniques, residues are coded using simple variations of still image compression algorithms. For example, MPEG, H.26x and many other video coding schemes use a spatial transform (such as the discrete cosine transform) for coding residues. Recently, the set partitioning in hierarchical tree (SPIHT) algorithm developed by Said and Pearlman [1993] [1996] by extending the embedded zerotree wavelet (EZW) algorithm originally introduced by Shapiro [1993] has been applied for dealing with residues. It achieves very good complexity-performance tradeoffs in the context of still image and video compression. The common part of the various EZW coding algorithms, even for residue compression, is a discrete wavelet transform (DWT). The idea of using transform coding for difference frames is based on the implicit assumption that prediction error signals (the difference frames or the residues) have spatial redundancy. However, it has long been noted that, contrary to the still image case, prediction image residues have little spatial structure, and there is no theoretical justification for involving a spatial transform. Hence in the proposed video coding system, the residues are coded without using any transforms.

In this thesis, the term "transform" means "spatial transform" and implies the transforms used in transform coding, such as DCT and DWT.

Besides the discrete wavelet transform, the EZW coders also consist of zerotree coding. Zerotree coding comes from the hierarchical structure of wavelet transforms. A zerotree is a data structure to be used to compress the wavelet coefficients. Zerotrees utilize the dependence of the wavelet coefficients corresponding to the same location but in different sub-bands. The tree structure can be defined as a group of pixels crossing different scales along a certain direction. A tree node, combined with all its descendants, forms a spatial orientation tree. Most of the coding efficiency of EZW results from spatially ordering the pixels by their magnitudes and effectively coding their coordinates. Because of the high compression performance of zerotrees, zerotree coding is applied for residues in the scheme proposed in this thesis.

Although video compression has inspired a great deal of research activity for both noiseless and noisy channels, the progress achieved for the noisy channel case is less significant than that for noiseless channel. For instance, the very good performances of EZW coders are obtained in error free channels. Unfortunately they are extremely sensitive to channel noise. To address this problem, sophisticated protection techniques have been presented, such as Redmill and Kingsbury [1996], Sherwood and Zeger [1997], and Man and Kossentini [1997]. The transmission data rate is inevitably increased in such schemes and additional computation is introduced. Therefore, the performance of the implemented scheme over a noisy channel without the aid of error correction or detection techniques should be seriously considered.

Summarizing the factors described above, this research is aimed to meet the following objectives:

- to design a coding scheme for prediction of image residues in video coding,
- to explore possibilities for channel error resilience and concealment,
- to develop a simple and efficient video coding scheme, which gives good performance over noisy channel conditions (such as weak radio links and noisy telephone lines), while maintaining the compression achievable over an error free channel.

To meet the above objectives, a complete video coding system is constructed. An efficient interframe coding scheme is designed for motion estimation and motion compensation. An effective Huffman coder is employed for the compression of motion vectors. For residue coding, which is the main objective of this research, zerotree coding is used without wavelet transforms. To switch between a spatial transform and a coder without a spatial transform to achieve maximum compression, spatial patterns of pixels are developed, which will occupy successive levels of the tree structure. With the spatial patterns, the standard quad-tree ordering of wavelet coefficient values is replaced by simple pixel values. For simulating transmission over noisy channels, a random error generator is applied.

This thesis includes 8 chapters. Besides the introductory chapter, Chapter 2 gives the necessary background and discusses the previous and current error control techniques for video coding. Chapter 3 describes the composition of the proposed video coding system and mainly explains the half-pixel block-matching technique. It also presents the results of motion estimation and motion compensation. Chapter 4 explains the patterns, the constraints for building patterns, the mechanism for adapting an image into zerotrees by using patterns, and the zerotree pattern residue coding. Chapter 5 describes the comparative tests on patterns and illustrates the performances of patterns. Chapter 6 provides compression results over error free channels and then compares the results with MPEG and those using DWT. Chapter 7 shows simulation results over noisy channels in different bit error rates and discusses the channel error resilience and concealment of the pattern zerotree residue coder and the wavelet zerotree residue coder. Chapter 8 concludes the thesis and makes recommendations for future work.

# Chapter 2

# Background

In this chapter, the well-known MPEG and H.261/H.263 video compression standards are reviewed as background knowledge. In addition, zerotree coding and some popular error control coding techniques for digital video are also included.

## 2.1 Studies on Moving Picture Coding

The problem of moving picture coding has inspired a great deal of research activity. Compression algorithms can be divided into lossless and lossy depending on the type of coding used. Lossless algorithms, such as entropy coding, use only statistical properties of data, remove less redundancy and therefore cannot achieve high compression ratios. However, lossless algorithms may be used for both preprocessing and post-processing in lossy algorithms. Standard lossy algorithms, such as H.261/H.263 and MPEG will be studied in this section.

### 2.1.1 The H.261/H.263 Compression Standard

H.261/H.263 is part of the H.324 video conferencing standard developed by the International Telecommunication Union (ITU). H.261/H.263 defines real time compression and decompression algorithms for video telecommunications. It is based on the 2-D Discrete Cosine Transform (DCT) with simple motion estimation between frames. The H.261 coding standard provides coded video at bit rates 64 Kbits/s and above, whereas the H.263 video coding standard provides coded video around 16 Kbits/s.

The block diagrams of the H.261/H.263 video encoder and decoder given by Furht. et al. [1997] are shown in Figure 2.1 and 2.2. respectively.

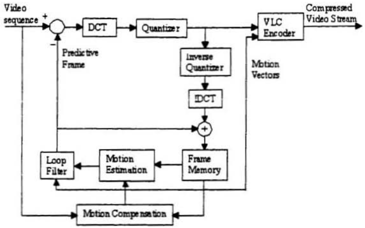


Figure 2.1 Block diagram of the H.261/H.263 video encoder.

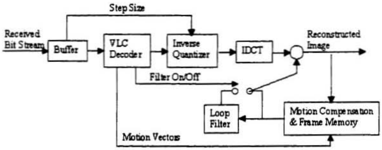


Figure 2.2 Block diagram of the H.261/H.263 video decoder.

Furht. et al. [1997] described H.261 supports low resolution formats like the Common Intermediate Format (CIF) whose resolution is 352 × 288 pixels and Quarter CIF (QCIF) whose resolution is 176 × 144 pixels. The maximum frame rate is 30 frames per second but it can be reduced depending on the application and bandwidth availability. However, H.263 has no bitrate limitation and is able to support five picture formats: CIF, QCIF, sub-QCIF, 4CIF, and 16CIF which actually gives a picture format of up to1408 × 1152 luminance resolution. Both in H.261 and H.263, color pictures are coded using one luminance, Y, and two color-difference components, $C_r$ and $C_b$.

There are usually two types of coded frames used in the H.261/H.263 standard: Intraframes (I-Frames) compressed using DCT, quantization, and variable-length coding (VLC) and Predictive Interframes (P-Frames) compressed using motion estimation and residue coding. As shown in Figure 2.1, the H.261/H.263 P-Frame compression algorithm is based on a motion-compensated interframe prediction and the DCT transform. The data is organized into macroblocks. Each macroblock is four 8 × 8 blocks of luminance, one of $C_r$ and one of $C_b$. The encoder begins by coding an intraframe macroblock and then sends it to the video multiplex coder. The first intraframe is then decompressed using the inverse quantizer and inverse DCT (IDCT), and then stored in the frame memory for interframe coding of the subsequence frames. During the interframe coding, each macroblock in the current frame is motion-predicted from the nearby macroblocks in the previous frame. The displacement between the previous and current macroblocks is considered as a motion vector for the macroblock in the current frame. If the prediction error is less than a certain threshold, no further processing is performed. Otherwise, the error is encoded using the same DCT and quantization. Finally, the motion vectors as well as the coded prediction errors are compressed with variable-length coding (VLC), such as Huffman coding to produce more compact data.

H.263 is an enhanced version of H.261, which achieves higher coding efficiency. The major improvements over H.261 consist of introducing a half-pixel motion compensation (like in MPEG-1 and 2) which allows more precise prediction. To allow further improved performance, the H.263 standard provides four negotiable options that can be used together or separately. Among these options, the utilization of syntax-based arithmetic coding instead of Huffman coding is available. Similar to MPEG standards, the H.263 coder can also generate B- Frames (see section 2.1.2.2), which are unavailable in H.261, but unlike MPEG, a B-Frame is coded together with a P-Frame as a single PB-Frame unit.

## 2.1.2 The MPEG Compression Standard

The *Moving Picture Experts Group*, from which the MPEG standard derives its name, is a working group operating within the International Standardisation Organization (ISO) and the International Electrotechnical Commission (IEC). Since starting its activity in 1988, MPEG has produced MPEG-1, MPEG-2, and MPEG-4 international standards and achieved great success.

### 2.1.2.1 Overview of MPEG Standards

The MPEG-1 standard formally known as ISO/IEC 11172 was originally intended for coding of video and associated audio at bit rates of up to 1.5 Mbit/s. MPEG-1 was originally aimed at digital storage, and in particular compact disks. The main applications targeted by MPEG-1 were prerecorded video, Interactive CD and games. Such applications do not require large image sizes and resolutions. The standard is in three parts: video, audio, and systems, where the last part gives the integration of the audio and video streams with the proper timestamping to allow synchronization of the two. Furthermore, the MPEG-1 video compression standard is the basis of the other MPEG higher version standards, and it might be compared with the proposed system, thus a detailed introduction on MPEG-1 is given in section 2.1.2.2.

MPEG-2 extends the functions provided by MPEG-1. It achieves efficient encoding of audiovisual information at a wide range of resolutions and bit rates, but also provides full interactive multimedia services.

The need for MPEG-3 was anticipated in 1991 for High Definition Television. However, it was discovered by late 1992 and 1993 that the MPEG-2 syntax simply scaled with the bit rate, so that the third phase was obviated.

The latest standard MPEG-4 was released in 1997. It is a standard for interactive networked multimedia incorporating both natural (pixel/sample based) audio and video and synthetic (2D and 3D animation based on polygons, splines, et al.) audio and video.

### 2.1.2.2 MPEG-1 Video Compression Standard

MPEG-1 was optimized to encode noninterlaced video at Source Intermediate Format (SIF) resolutions ($352 \times 240$) at 30 frames per second, or $352 \times 288$ at 25 fps. The bit rate required for these resolutions is about 1.2 Mbit/s. Adding two channels of audio information increased the bit rate to 1.5 Mbit/s. Also, MPEG-1 allows the encoding of much larger picture sizes and correspondingly higher bitrates. If the images are in color, they should be converted to YUV space, and the two chrominance channels (U and V) are decimated further to half size.

MPEG-1 makes use of a number of different techniques to achieve its high compression ratio. In the same way as H.261, MPEG-1 exploits both the spatial and temporal redundancies of motion pictures. Figure 2.3 and Figure 2.4 given by Furht, et al. [1997] illustrate the block diagram of MPEG-1 encoder and decoder.

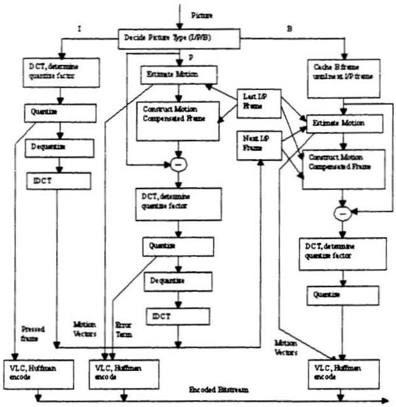


Figure 2.3 Block diagram of the MPEG-1 encoder.

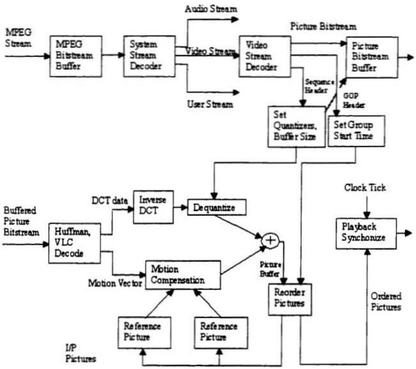


Figure 2.4 Block diagram of the MPEG-1 decoder.

Gall [1991] pointed out that two basic techniques were used in the MPEG-1 video compression algorithm:

- block-based motion compensation for the reduction of the temporal redundancy,
- transform domain – Discrete Cosine Transform (DCT) based compression for the reduction of spatial redundancy.

Next, the MPEG-1 algorithm will be explained from these two aspects.

**Temporal Redundancy Reduction**

There are three types of coded frames in MPEG-1: Intraframes (I), Predicted frames (P), and Bidirectional frames (B). An intraframe is simply a frame coded as a still image, not using any past history. A predicted frame is coded with reference to the most recently reconstructed I or P frame and will be used as a reference for future P-Frames. A bidirectional frame is predicted from the closest two I or P frames, one in the past and one

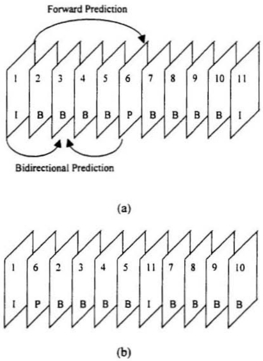


Figure 2.5 MPEG-1 frames where (a) is an example of MPEG-1 frame structure and (b) is the MPEG-1 frames sequenced in compressed order.

in the future. Unlike P-Frames and I-Frames, B-Frames, which provide the highest amount of compression, are never used as references for prediction. The organization of the three frame types in MPEG-1 is quite flexible. It depends on application-specific parameters. Figure 2.5 (a) gives an example of MPEG-1 frames. The first frame in the sequence is an I-Frame, followed by four B-Frames; the sixth frame is a P-Frame. Because both the first frame (I-Frame) and the sixth frame (P-Frame) are used to compress the B-Frames, the I-Frame and the P-Frame should be sent before the four B-Frames. Hence, the transmission of such MPEG compressed video frames is in a different order, as shown in Figure 2.5 (b). In the receiver, we first decode the I-Frame, then decode the P-Frame, keep both of those in memory, and then decode the four B-Frames, and so on.

In all cases when a frame is coded with respect to a reference, motion compensation is applied for both causal prediction of the current frame from a previous frame and non-causal interpolative prediction from past and future frames, to improve the coding efficiency.

Actually, the compression of the frames is conducted in sub-units of macroblocks, which are the motion compensation units. Similar to the H.261/H.263 stand, a macroblock is $16 \times 16$ of luminance (Y) channel and the corresponding $8 \times 8$'s in the two chrominance channels (U and V), and motion prediction is performed in the luminance (Y) channel on $16 \times 16$ blocks. For instance, in the more general case of a bidirectionally coded frame, the B-Frame to be encoded is subdivided into the $16 \times 16$ macroblocks, and each $16 \times 16$ B-macroblock can be one of four types: Intrablock, Forward-Predicted block, Backward-Predicted block and Average block. Gall [1991] listed the prediction modes for a macroblock in the current B-Frame, as shown in Table 2.1. In Table 2.1, $\bar{x}$ is the coordinate of the picture element, $\overline{mv_{01}}$ is the motion vector relative to the forward reference frame $I_0$. $\overline{mv_{21}}$ is the motion vector relative to the backward reference frame $I_2$ and $I_1$ is the current B-Frame. The prediction errors (residues) are the difference between this macroblock and the closest match. Table 2.1 illustrates that the expression for the predictor for a given macroblock not only depends

Table 2.1 Prediction Modes for Macroblock in B-Frame

| **Macroblock Type** | **Predictor** | **Prediction Error** |
|---|---|---|
| Intra | $\hat{I}_1(\bar{x}) = 128$ | $I_1(\bar{x}) - \hat{I}_1(\bar{x})$ |
| Forward Predicted | $\hat{I}_1(\bar{x}) = \hat{I}_0(\bar{x} + \overline{mv_{01}})$ | $I_1(\bar{x}) - \hat{I}_1(\bar{x})$ |
| Backward Predicted | $\hat{I}_1(\bar{x}) = \hat{I}_2(\bar{x} + \overline{mv_{21}})$ | $I_1(\bar{x}) - \hat{I}_1(\bar{x})$ |
| Average | $\hat{I}_1(\bar{x}) = \frac{1}{2}[\hat{I}_0(\bar{x} + \overline{mv_{01}}) + \hat{I}_2(\bar{x} + \overline{mv_{21}})]$ | $I_1(\bar{x}) - \hat{I}_1(\bar{x})$ |

on reference frames (past and future) but also depends on the motion vectors, and motion information can be represented by one or two motion vectors per 16 × 16 sub-block of the picture depending on the type of motion compensation: forward-predicted, backward-predicted, or average. It also indicates the I-macroblock and the P-macroblock cases, where the former should be the Intrablock type and the later should be the Forward-Predicted block type.

Although the MPEG-1 standard does not specify how to calculate these motion vectors, block-matching motion estimation techniques are likely to be used because of the block-based motion representation. In the decoder, these motion vectors and the associated prediction error signals are applied back to the previous and following frames to recover the original data.

**Spatial Redundancy Reduction**

MPEG-1 considers that prediction errors, like a still image, have a high spatial redundancy. Thus, as in JPEG (Joint Photographic Experts' Group) and H.261, the DCT algorithm can be applied to the prediction errors to increase the compression through the removal of spatial correlation. Furht, et al. [1997] studied the theory of DCT coding. They stated that the discrete cosine transform (DCT) was used to decorrelate the pixel or

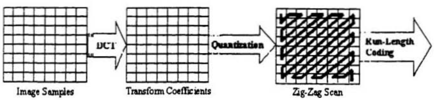


Figure 2.6 Residue coding with DCT.

error values, thereby storing the bulk of the image data in the fewest number of terms. Specifically, the DCT's are done on $8 \times 8$ blocks. Furht, et al. [1997] also explained that there are three stages in the DCT residue coding process, as shown in Figure 2.6:

1) *Computation of the Transform Coefficients*

In this stage, the DCT concentrates the representation energy of each block by converting a block of pixels into a block of transform coefficients of the same dimensions.

Equation 2.1 shows the two dimensional DCT of an $N \times N$ pixel block, where

$$F(u,v)=\frac{2}{N}C(u)C(v)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}f(i,j)\cos(\frac{(2i+1)u\pi}{2N})\cos(\frac{(2j+1)v\pi}{2N}). \qquad (2.1)$$

where,

$$C(x)=\begin{cases}\frac{1}{\sqrt{2}} & ,x=0\\ 1, otherwise\end{cases}$$

$f(i,j)$ represents the pixel values and $F(u,v)$ the transform coefficients.

These coefficients represent the spatial frequency components of the original pixel block. Such a transform does not produce any compression. However, it clusters the majority of the coefficients around the DC value, since in a typical image, the blocks of pixels tend to have little high frequency content and most of the energy tends to be stored in the low frequency coefficients. The lowest frequency coefficient is in the upper left hand corner of a block, and the highest frequency coefficient is in the bottom right. This clustering effect or energy compaction is exploited later by the quantization.

*2) Quantization of the Transform Coefficients*

The DCT coefficients are quantized so that the non-significant coefficients are set to zero and the remaining ones are represented with a reduced precision. This is achieved by dividing each of the coefficients by an integer taken from a quantizer scale and then rounding the result to the nearest integer. The quantizer scale depends on the coefficient being quantized. Since the human visual system (HVS) is mostly affected by the low frequency coefficients, the quantizer scale for these is usually taken to be smaller than those for the high frequency coefficients. This results in loss of information but also in compression since most of the quantized coefficients in a block are zero.

*3) Reorganization and Compression of the Quantized Data*

At the last stage, first, the resulting block after quantization is reorganized into a zig-zag pattern. This sequencing acts to group the low frequency terms together, and as a result also groups the high frequency terms, many of which were zeroed out by quantization. These groups of zeros are then subjected to run-length coding to further increase compression.

After reorganization, the resulting pattern of symbols obtained from the zig-zag sequence are coded using a variable length coding (VLC) technique. Variable length coding represents integer values by a length field followed by the number of bits needed to represent the value. A Huffman table for the DCT coefficients is used to code events. Only those events with a relatively high probability of occurrence are coded with a variable-length code; the less-likely events are coded with an escape symbol followed by fixed length codes, to avoid extremely long code words and reduce the cost of implementation.

As described by Furht, et al. [1997], the run-length of zeros preceding a non-zero coefficient and the VLC length field are combined to form a single symbol for entropy coding. Entropy coding, such as Huffman coding, is a statistical coding technique that assigns codewords to values to be encoded. Values that occur frequently are given short codewords, and those that occur infrequently are given longer ones.

In summary, the basic scheme of MPEG-1 compression is to predict motion from frame to frame in the temporal direction, and then to use DCT's to organize the redundancy in the spatial directions.

## 2.2 Zerotree Coding

As introduced in section 2.1, the motion-compensated difference frames (residues) are usually coded using simple variations of still image compression algorithms, such as a transform coder in MPEG and H.261/H.263. Zerotrees arise from the transform coding. Zerotree based algorithms, such as various embedded zerotree wavelet (EZW) image coding algorithms, represent the state-of-the-art in wavelet transform based image coding, such that transform coding with zerotrees is usually called zerotree transform coding.

### 2.2.1 A Typical Transform Coding Scheme

There are three basic components in a generic transform coder: a transformation, a quantizer, and data compression, as shown in Figure 2.7 given by Shapiro [1993].

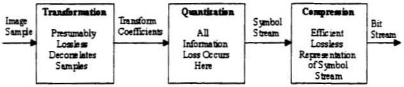


Image Sample
Transformation
Presumably Lossless Decorrelates Samples
Transform Coefficients
Quantization
All Information Loss Occurs Here
Symbol Stream
Compression
Efficient Lossless Representation of Symbol Stream
Bit Stream

Figure 2.7 A generic transform coder.

The original image is passed through some transformation, such as the discrete cosine transform (DCT) or the discrete wavelet transform (DWT), to produce transform coefficients. Such transformation is presumably lossless. After the transformation, the coefficients are quantized to generate a symbol stream. All information loss occurs here. Then, the symbol stream is represented by the bit stream to improve transmission efficiency.

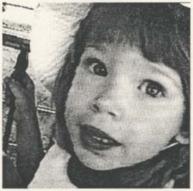
### 2.2.2 DCT and DWT

The discrete cosine transform has been widely used in video coding, such as in MPEG and H.261/H.263 (section 2.1). The DCT coder decomposes images into representations where each coefficient corresponds to a fixed-size spatial area and frequency band. The formula for computing DCT coefficients has been given in section 2.1. The goal of the transformation is to produce decorrelated coefficients, which allow using scalar quantizers efficiently. The drawback of DCT is that the edge information requires many nonzero coefficients to represent sufficiently.

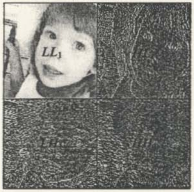
Shapiro [1993] stated that wavelet techniques offer benefits at low bit rates since information at all scales is available for edges and regions. The discrete wavelet transform decomposes an image into a set of sub-images called shapes with different resolutions corresponding to different frequency bands.

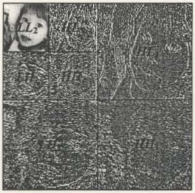
The original image is typically decomposed into four sub-bands using separable filters: one that has been lowpass filtered in both vertical and horizontal directions (*LL*), one that has been highpass filtered in the vertical and lowpass filtered in the horizontal (*HL*), one that has lowpass filtered in the vertical and highpass filtered in the horizontal (*LH*), and one that has been highpass filtered in both directions (*HH*). Each coefficient represents a spatial area of the original image. The decomposition process can be carried out on the lowpass-lowpass version of the image a number of times depending on the scale of compression required.

Figure 2.8, given by Umbaugh [1998], illustrates the results of applying the wavelet decomposition to an image, where the sub-bands labeled $LH_1$, $HL_1$, and $HH_1$ represent the finest scale wavelet coefficients and $LH_2$, $HL_2$, and $HH_2$ the second scale coefficients. To obtain the next coarser scale of wavelet coefficients, the lowest frequency sub-band $LL_2$ can be further decomposed until some final scale is reached. Note that for each coarser scale, the coefficients represent a larger spatial areas of the image but a



(a)





(b) (c)

Figure 2.8 Wavelet decomposition where (a) is the original image, (b) is a one-scale decomposition, and (c) is a two-scale decomposition.

narrower band of frequencies. For example, each coefficient in Figure 2.8 (a) represents a spatial area corresponding to approximately a $2 \times 2$ area of the original image, and each coefficient in the sub-bands $LL_2$, $LH_2$, $HL_2$ and $HH_2$ in Figure 2.8 (b) represents a $4 \times 4$ area of the original image. Also we can see the *LL* image in the upper-left corner, the *LH* images on the diagonals, and the *HH* in the lower-right.

Numerous filters can be used to implement the wavelet transform, such as the Daubechies and the Haar. According to Shapiro [1993], the filters used to compute the discrete wavelet transform in his coding simulations are based on the 9-tap symmetric quadrature mirror filters (QMF) due to their good localization properties and their symmetry.

The inverse wavelet transform is performed by enlarging the wavelet transform data to its original size. Decoding then inserts zeros between each value, convolves the corresponding (lowpass and highpass) inverse filters to each of the four sub-images, and sums the results to obtain the original image.

### 2.2.3 Zerotrees in EZW Coder

It has long been noticed that in most natural images, whose main features are associated with edges and textures, the image's energy is concentrated in the low frequency components and the variance decreases as moving from the highest to the lowest scales of the wavelet sub-band pyramid. That is to say, there are large areas in the high-frequency sub-bands of the wavelet transform where coefficients are insignificant and hence can be coarsely quantized (mostly to zero). If there is a way to efficiently predict these patches of insignificant coefficients from other coefficients, which are already quantized and transmitted, great savings can be achieved. Zerotree coding is such a way to improve the compression of wavelet coefficients. Shapiro [1993] stated that zerotrees allow the successful prediction of insignificant coefficients across scales to be efficiently represented as part of exponentially growing trees.

A zerotree is a data structure invented by Shapiro. According to Shapiro [1993], a wavelet coefficient $x$ is said to be *insignificant* with respect to a given threshold $T$ if $|x| < T$, and the zerotree is based on the hypothesis that if a wavelet coefficient at a coarse scale is insignificant with respect to a given threshold $T$, then all wavelet coefficients of the same orientation in the same spatial location at finer scales are likely to be insignificant with respect to $T$. This hypothesis is often true by empirical evidence.

Next, Shapiro's embedded zerotree wavelet (EZW) image coding algorithm will be introduced in detail.

The EZW image coding algorithm utilizes not only the decorrelating properties of the wavelet transform, but also the dependence of the wavelet coefficients corresponding to the same spatial location but in different sub-bands. There is a spatial self-similarity between sub-bands (refer to Figure 2.8 in section 2.2.2). This spatial relationship is naturally represented as a tree. For example, in a hierarchical wavelet decomposition, except the highest frequency sub-band, every coefficient at the coarse scale (*parent*) can be related to a set of coefficients at the next finer scale of similar orientation (*children*).

Figure 2.9 shows how the tree is defined in a pyramid constructed with recursive four sub-band splitting. The arrow points from the sub-band of the parents to the sub-band of the children. Note that for the lowest frequency sub-band, each parent node has three children; however, for the other sub-bands, each parent has four children. For a given parent, the set of all coefficients at all finer scales of similar orientation corresponding to the same location are called *descendants*, whereas the set of all coefficients at all coarser scales of similar orientation corresponding to the same location are called *ancestors*. Based on the hypothesis described above, whenever a parent node has a small value, it is likely that child nodes are also small.

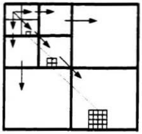


Figure 2.9 Parent-child dependencies of sub-bands.

A *zerotree* represents the insignificance information in a given orientation over an approximately square spatial area at all finer scales up to and including the scale of the zerotree root. A scanning of the coefficients is performed in such a way that each coefficient within a given sub-band is scanned before any coefficient in the next sub-band. To determine if a coefficient $x$ is an element of a zerotree, a threshold $T$ is given. If $x$ and all of its descendents are insignificant with respect to $T$, $x$ is said to be an element of a zerotree for threshold $T$. In practice, four symbols are used to represent the significance maps of wavelet coefficients:

- Zerotree Root: An element of a zerotree for threshold $T$ is a zerotree root if it is not the descendant of a previously found zerotree root for threshold $T$.
- Isolated Zero: Isolated zero means that the coefficient is insignificant but has

some significant descendants.

- Positive Significant
- Negative Significant

The flow chart, given by Shapiro [1993] for the decisions made at each coefficient, is shown in Figure 2.10. Only the coefficients labelled with the above four symbols are coded. A zerotree root is encoded with a special symbol indicating that the insignificance of the coefficients at finer scales is completely predictable.

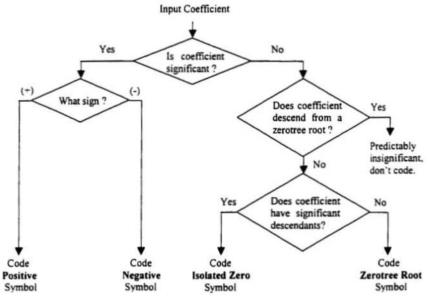

Input Coefficient
Is coefficient significant ?
Yes
No
What sign ?
(+)
(-)
Does coefficient descend from a zerotree root ?
Yes
Predictably insignificant, don't code.
No
Does coefficient have significant descendants?
Yes
No
Code **Positive** Symbol
Code **Negative** Symbol
Code **Isolated Zero** Symbol
Code **Zerotree Root** Symbol

Figure 2.10 Flow chart for encoding a coefficient of the significance map.

The implementation of the EZW algorithm includes two passes: a dominant pass where significant coefficients are identified and a subordinate pass where such coefficients are refined. Note that the threshold $T$ is renewed at each routine. The processing continues alternating between dominant and subordinate passes and can stop at any time. It results in a progressive transmission property, which is a very attractive feature in many applications. The saving in bitrate comes partially from the predicted quantizations and partially from the highly nonuniform conditional probability distribution of the zerotrees, which is exploited in the arithmetic coder.

### 2.2.4 Various EZW coders

Following the original Embedded Zerotree Wavelet (EZW) image coding algorithm developed by Shapiro in 1993, the EZW structure has been extensively studied in the context of still image and video compression. Various EZW coders have been developed. They present a progressive transmission scheme that orders the coefficients by magnitude and transmits the most significant bits first because they have a larger content of information.

The common part of the various EZW image coding algorithms is a pyramid wavelet sub-band decomposition of the original image which has been introduced in the last section, and the main difference is the way the significance map is coded during the dominant pass in a zerotree.

Among variations of Shapiro's algorithm, the most notable is the Set Partitioning In Hierarchical Trees (SPIHT) algorithm developed by Said and Pearlman [1993] [1996]. The SPIHT algorithm employs a set partitioning sorting algorithm to code the significance map very efficiently. It yields very good performance-complexity tradeoffs in the context of still image and video compression for noiseless channels. The SPIHT not only outperforms other EZW algorithms but also provides an alternative explanation and implementation of the EZW principles. Following is a brief explanation of SPIHT.

In SPIHT, a tree structure, called spatial orientation tree, defines the spatial relationship on the hierarchical sub-bands. Figure 2.11 shows an example of parent-offspring dependencies in the spatial orientation tree where the arrow points from the parent node to its four children. The tree is defined in such a way that each node has either no children (the leaves) or four children, which always form a group of 2 × 2 adjacent pixels. The pixels in the upper-left corner are the tree roots and are also grouped in 2 × 2 adjacent pixels. In each group of the tree roots, one of them (indicated by the star) has no descendants and is called the star root. Except for the star roots, every other roots has four child nodes, every child nodes also has four children, and so on. The tree grows in this way until it reaches the leaves.

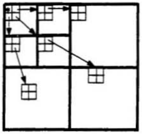


Figure 2.11 Parent-offspring dependencies in the orientation tree.

The algorithm is also performed through the dominant and the subordinate pass at each threshold as the EZW algorithm but in a different way. In each pass, every node and its descendants are compared to a threshold, and a state (symbol) is assigned to each node. Different from the EZW algorithm, there are four states used for each threshold level:

- IT (insignificant tree): compared with the threshold, both the node and all its descendants are insignificant.

- IR (insignificant root): compared with the threshold, the node is insignificant but not all its descendants.
- SR (significant root): compared to the threshold, the node is significant but all its descendants are insignificant.
- ST (significant tree): compared to the threshold, both the node and some of its descendants are significant.

From one threshold level to another, the state of a node changes. The state transition diagram is shown in Figure 2.12.

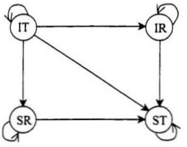


Figure 2.12 State transition diagram.

The SPIHT algorithm is described below:

1) Define initial threshold: Set threshold $T$ to the highest power of two less than the maximum absolute value of all wavelet coefficients $c_n$ and output $k$ to the decoder, where

$$k = \lfloor \log_2(\max_n |c_n|) \rfloor$$

$$T = 2^k.$$

2) Initialize index lists: All the tree roots are put into the dominant list. The subordinate list is initialized to an empty list.

3) Initialize states: Set $S_{\text{old}} = \text{IT}$ for all nodes.

4) Dominant pass: For each entry $n$ in the dominant list do

   a) Save the old state $S_{\text{old}}$ and find the new state $S_{\text{new}}$ at the current threshold, using the definition of the states above.

   b) Output the code for state transition $S_{\text{old}} \rightarrow S_{\text{new}}$.

   c) If there is no change in state ($S_{\text{old}} = S_{\text{new}}$), then goto step 5).

   d) If $S_{\text{old}} \neq \text{SR}$ and $S_{\text{new}} \neq \text{SR}$ (the node has just become significant) then add index $n$ to the subordinate list (for the next run) and output the sign of $c_n$.

   e) If $S_{\text{old}} \neq \text{IR}$ and $S_{\text{new}} \neq \text{SR}$ (the children of the node have just become significant), add the indices of the immediate children of the node to the dominant list.

   f) If $S_{\text{new}} = \text{ST}$ then remove the index $n$ from the dominant list.

5) Subordinate pass: For all entries in the subordinate list, output the next (i.e., $(k-1)$-th) most significant bit.

6) Threshold update: Set $\frac{T}{2} \rightarrow T$, decrement $k$ and goto step 4).

In order to improve the efficiency of the algorithm, entropy coding can be performed with an adaptive arithmetic coder.

## 2.3 Error Control Coding for Video

Many video compression algorithms employ block-based techniques for their implementation, such as MPEG, H.261/H.263, and others. The input signal is split into blocks. After compression, variable-length coded data is made for each block. If these variable-length blocks are transmitted consecutively through channels, the propagation of channel errors may highly affect the compressed data. The basic reason is that channel errors cause a non-recoverable loss of synchronization with the start and the end of blocks between the encoder and decoder. Even if the synchronization can be regained after the loss of a few blocks, the following data can not be decoded at the right location. For video coding, this can result in large areas of the picture being spatially displaced, and then leads to an uncontrolled degradation in recovered image sequence quality.

According to Redmill and Kingsbury [1993], channel errors can propagate within the decoder and degrade its performance in several ways:

- Bit decoding: In variable length code-word schemes, channel errors can effect the decoding of following code-words.
- Code-word decoding: In most schemes the meaning of individual code-words is dependent on previous code-words.
- Spatial domain: In most sub-band schemes, such as EZW and SPIHT codec, a single error in a sub-band will affect a local region of the picture (not just one pixel).
- Temporal domain: In systems that employ temporal prediction, such as motion estimation and motion compensation for interframe coding, errors in one decoded frame will propagate to all dependent frames.

To address this problem, numerous sophisticated techniques have been developed over the last several decades.

A traditional approach to protect source coder information from the effects of a noisy channel is to include unique synchronization words followed by some block address information at regular points. This makes it possible to regain synchronization after errors and decode the following blocks of data at the right location. However, this approach also adds redundancy information to the code and thus degrades the performance in error free channels. In order to maintain low redundancy, the synchronization words must be used relatively infrequently; thus the resulting scheme performs reasonably at moderate bit error rates (BER), but deteriorates rapidly at high bit error rates (BER). This type of technique was used in Ferguson and Rabinowitz [1984] and Montgomery and Abrahams [1986].

Another approach is error correction coding (ECC). which can be applied alone or together with synchronization words or other techniques. The error correction techniques also involve the addition of extra redundancy information. They use Cyclic Redundancy Codes (CRC) or parity codes to check if the received data is correct. The simulation results obtained by MacDonald [1992] illustrate that ECC works well if the BER is known in advance to be fairly small and constant but not well for channels with variable BER or long bursts.

A relatively new approach is to use custom-designed schemes such as the error resilient entropy code (EREC) method introduced by Cheng and Kingsbury [1992] and Redmill and Kingsbury [1993] [1996]. Cheng and Kingsbury [1992] described that the EREC is a development of the error resilient positional code (ERPC) for adapting existing entropy coding schemes to give increased error resilience. It is primarily concerned with minimizing the first two forms of channel error propagation. The EREC algorithm works by reorganizing the variable-length blocks such that each block starts at a known position within the code, which means that the decoder is automatically synchronized at the start of each block. It gives increased priority to important motion vector and low frequency information over less critical high frequency information. In Redmill and Kinsbury [1993][1996], they applied the EREC to the H.261 video coding scheme and a typical still image coding scheme using DCT and variable-length coding. The results show that the

EREC can significantly reduce the channel error propagation effects and the redundancy involved is less than that of using synchronization words. By involving more complicated hybrid schemes, the EREC method does not require that the overall code must be of fixed length per frame any more. However, it is very complex to process.

# Chapter 3

# System Overview

Although my research focuses on coding motion picture residues, a complete video coding system was to be constructed for simulation and evaluation.

## 3.1 System Diagram

The implemented video coding system, as shown in Figure 3.1, mainly includes three parts: motion estimation and motion compensation, motion vector encoder/decoder, and residue encoder/decoder.

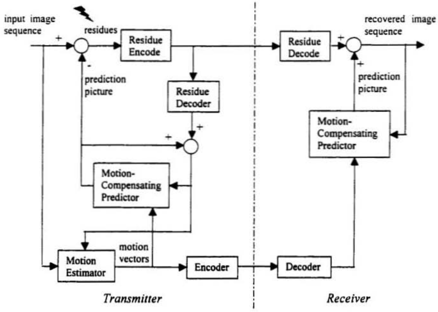


Figure 3.1 Diagram of the proposed system.

It works in this way: the input image sequences are separated into blocks of equal sizes; by block-searching between each previous frame and the current frame, motion vectors are obtained for the blocks in the previous frame. The motion vectors represent the motion information of a frame. By shifting the blocks in the previous frame according to the motion vectors, the prediction picture of the current frame is obtained. The difference between the prediction picture and the current frame is called the residue. Apart from the first frame, the compressed motion vectors and residues need to be sent to the receiver. In the receiver, starting from a previously recovered picture, the decoder shifts the blocks according to the motion vectors, then adds the residue to obtain the current reconstructed picture.

Half-pixel accuracy motion estimation and motion compensated techniques are designed for interframe coding. Huffman coding/decoding is applied for coding motion vectors. Zerotree pattern coding/decoding is developed for residue coding, which is the main part of the thesis and will be introduced in chapter 4.

## 3.2 Motion Estimation and Motion Compensation

The interframe estimation used in the system is a half-pixel block-matching technique. The block-matching theory assumes that a picture is composed of rigid objects in translation. This is based on the fact that a complex motion can be decomposed into a sum of translated components. The aim of block-matching techniques is to find the best block correspondence between two successive frames (such as a current frame and a previous frame) according to a criterion, generally the minimization of the mean square error (MSE) or mean absolute error (MAE) between the two blocks.

The MSE can be calculated with

$$d_{mse}(M_i, v) = \frac{1}{w_b h_b} \sum_{x \in M_i} (I_c(x) - I_r(x - v))^2 \qquad (3.1)$$

which is a function of the block, $M_i$, and the motion vector, $v$. The current frame and the reference frame are represented by $I_c$ and $I_r$, respectively; the width of the block, $w_b$ and the height of the block, $h_b$, are also required.

The MAE can be calculated with

$$d_{mae}(M_i, v) = \frac{1}{w_b h_b} \sum_{x \in M_i} | I_c(x) - I_r(x - v) | \qquad (3.2)$$

which is also a function of the block, $M_i$, and the motion vector, $v$.

The selection of the best prediction block is based on an exhaustive search of all possible candidates. Once the matching blocks in the previous frames are found, the same motion vector is assigned to each pixel of the same block.

The conventional closest-pixel block-matching method can only estimate displacement to the nearest pixel. However, it is possible to interpolate the pixels on a four times grid around the search area, and then the block can be displaced by fractions of half a pixel. Figure 3.2 illustrates the half-pixel method, where the light black rectangle in the previous frame is the best prediction block after the closest-pixel block-matching and the heavy black rectangle represents the best matching block after the half-pixel block search, which is the final result.

In the simulation simulations in this thesis, the size of the segmented blocks is 16 $\times$ 16 pixels, and the search range is $\pm$ 14 pixels. In order to save transmission bits used for motion vectors and to speed up the program, a threshold $T$ is introduced to the block-

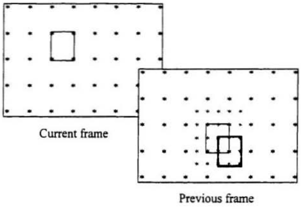


Figure 3.2 Half-pixel accuracy block-matching technique.

matching method. For each block in the current frame, before beginning the block-matching search, the MAE with the block at the same location in the previous frame is tested to find out whether or not it is smaller than a prespecified threshold $T$. In this case, it is considered that there is no motion, and the motion vector is 0. No further block-matching is applied. The threshold used in the simulations is 250. Note that this technique is also used in MPEG.

Figure 3.3 illustrates a video sequence, called *Football*. This video shows a football game, and contains lot of motion. Figure 3.4 gives the motion estimation and motion compensation pictures with this sequence. The slashes and crosses of Figure 3.4 (f) are motion vectors that represent the real motion between the previous frame and the current frame, where a slash indicates the displacement (twice of the real displacement) of a block and a cross means there is no movement for this block. Comparing the difference between the two successive frames and the difference between the current frame and the prediction picture, we can see the effect of the motion compensation, since the latter picture contains less difference in gray levels. Boundaries of the blocks are clearly visible on the residue picture, especially the boundaries of the moving blocks. In the decoder, the residue is added to the prediction picture to obtain the recovered picture.

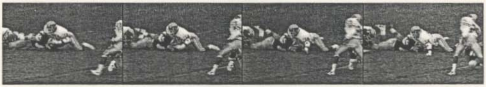


Figure 3.3 *Football* video sequence.

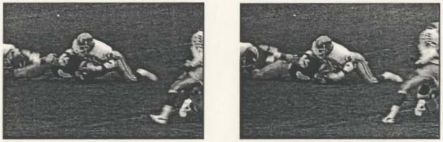
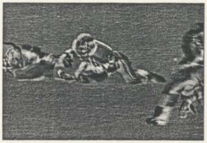
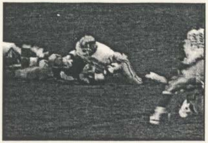

(a) Previous frame (Figure 3.3 Frame1)
(b) Current frame (Figure 3.3 Frame2)

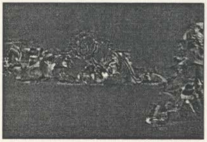
Figure 3.4 Simulation results of motion estimation and motion compensation for the *Football* video (352 × 240, 8 bits/pixel, 25 frames/s).



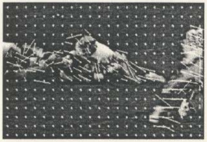
(c) Difference between (a) and (b)



(d) Prediction picture



(e) Residue picture (difference between (a) and (d))



(f) Motion vectors

Figure 3.4 Simulation results of motion estimation and motion compensation with the *Football* video (352 × 240, 8 bits/pixel, 25 frames/s). (Continued)

## 3.3 Coding of Motion Vectors

There are three steps for coding motion vectors:

- Concatenate the motion vectors obtained from each frame in one file.
- Find the difference between successive motion vectors.
- Use Huffman coding to code these difference data.

The Huffman code was developed by D. Huffman in 1952. Huffman coding is the most popular entropy coding scheme. The Huffman coding algorithm can be explained in four steps:

1) Find the probability of occurrence for each symbol in the alphabet.
2) Arrange the symbol probabilities in descending order and consider them as leaf nodes of a tree.
3) If there is more than one node,
    a) Merge the two nodes with smallest probabilities to form a new node whose probability is the sum of the two merged nodes.
    b) Arbitrarily assign 1 and 0 to each pair of branches merging into a node.
    c) Goto step 2).
4) Read sequentially from the root node to the leaf node where the symbol is located.

Thus, in Huffman coding, those symbols with the highest probability of occurrence at each step are given the smallest bit string representations. This ensures that the smallest possible number of bits will be used to represent a statistically likely word in the vocabulary.

The preceding algorithm gives the Huffman code book (lookup table) for any given set of probabilities. Huffman coding and decoding is conducted simply by looking up values in a table.

# Chapter 4

# Pattern Zerotrees Design

In the proposed video coding scheme, the difference frames (residues) are coded by a zerotree pattern coder. In order to use zerotrees (zerotree coding) without any transforms, ordering patterns are designed for switching between a spatial transform and a residue coder without spatial transform.

## 4.1 State-of-the Art Zerotree Wavelet Coding

Typically, a zerotree wavelet coder includes two main parts: one is the discrete wavelet transform, the other is zerotree coding. It is the zerotrees that realize large compression.

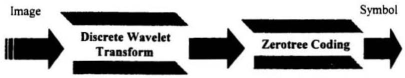


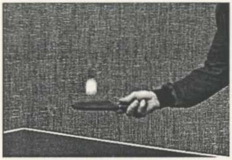
Figure 4.1 A zerotree wavelet coding scheme.

As introduced in section 2.2.2, the wavelet transform has a subband tree structure; a result of down-sampling at each subband stage. The zerotree wavelet coder utilizes the decorrelating properties of the transform to define a zerotree that allows the efficient use of scalar quantizers.

Since prediction image residues have little spatial content, a simple residue coding scheme is developed. The suggested residue coding scheme utilizes an ordering pattern to replace the discrete wavelet transform part in Figure 4.1.

## 4.2 8 × 8 Ordering Patterns

To illustrate that zerotree coding can be directly applied to pixels rather than to wavelet coefficients, I present some simulations where the DWT of Figure 4.1 is simply removed, and the zerotree structure is overlaid on the original image without consideration of matching the distribution of pixels to the zerotree structure. I call this method "Raw Zerotree Coding". For error-free channels, frame 10 and frame 11 of *Table Tennis* are used for the simulation. Figure 4.2 shows the recovered frame 11 where residues are coded by either raw zerotree coding or zerotree wavelet coding at bit rate of 0.1 bpp. From the results, it can be seen that good compression performance is gained


(a) PSNR = 36.09 dB

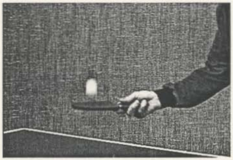

(b) PSNR = 36.61 dB

Figure 4.2 Recovered frame11 for *Table Tennis* where residues are coded by (a) raw zerotree coding or (b) zerotree wavelet coding.

with the raw zerotree coding because of the use of zerotrees, but the better compression performance is obtained with DWT. It will be demonstrated in section 6.3.2 that with ordering patterns, the same compression performance can be achieved as that with DWT. On noisy channels, the performance of raw zerotree coding is discussed in section 7.1.2. It is also explained by the results in section 7.1.2 why raw zerotree coding should not be applied for dealing with residues.

In short, the above analysis illustrates that the DWT is not essential for residue coding, but just feeding the picture straight into the zerotree coder is not an optimal strategy. A method is needed to organize pixels into successive levels of zerotrees following certain principles. In the thesis, $8 \times 8$ patterns and $16 \times 16$ patterns are designed for this purpose.

An $8 \times 8$ pattern is a block with $8 \times 8$ pixels. These pixels are grouped according to the quad-tree structure. A quad-tree is a tree structure such that each node, except the bottom nodes (the leaves), has four child nodes. In this way, there are hundreds and thousands of possibilities to build such patterns. To reduce the possibilities, a set of constraints for building patterns is predefined.

### 4.2.1 Constraints for Building 8 × 8 Patterns

The motion-compensated difference picture, unlike the still image, has a lot of pixels with zero gray values. Because of the block-based nature of the motion-compensation process, edges of the blocks, especially the moving blocks, are clearly visible on a residue picture. That means higher-frequency information concentrates on the edges of a block, and correspondingly lower-frequency information in the center of a block. So the tree is defined to grow out from the center of an 8 × 8 block.

Moreover, the number of pixels in a block and the number of nodes in a quad-tree should be considered. Figure 4.3 shows an example of an 8 × 8 block with 64 pixels and a 2-level quad-tree with 21 nodes.

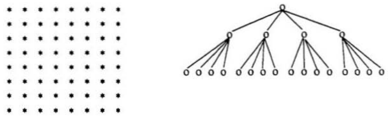

(a) (b)

Figure 4.3 An 8 × 8 block (a) and a 2-level quad-tree (b).

Generally, a block has $2^n \times 2^n$ pixels, where $n = 1,2,3,\cdots$; a $k$-level ($k = 1,2,3,\cdots$) quad-tree has $m$ nodes, where $m = 1 + \sum_{i=1}^{k} 4^i$. To arrange the $2^n \times 2^n$ pixels in several quad-trees, equation (4.1) is obtained as

$$am + b = 2^n \times 2^n, \tag{4.1}$$

where $a$ and $b$ are integers.

For an 8 × 8 block ($n$ = 3) and no pixel left ($b$ = 0) case, equation (4.1) is rewritten as

$$am = 64 \tag{4.2}$$

$$m = \frac{64}{a}. \tag{4.3}$$

Because m is an odd integer and m > 1, but $64/a$ is either an even integer or a fraction, hence equation (4.3) is untenable. That means it is impossible to arrange an 8 × 8 block of pixels in quad-trees with no pixels left over.

For an 8 × 8 block ($n = 3$), considering 1 pixel left ($b = 1$) case,

$$am + 1 = 64. \tag{4.4}$$

One solution of the above equation, gives $m = 21$, $a = 3$. That means the 63 pixels can be put in three 2-level quad-trees.

Therefore, an 8 × 8 pattern should have a root which has no child node.

In addition, to improve the compression of a tree structure, the nodes that have similar status should be arranged as close to each other as possible and if possible have the same parent and be located in the same level.

On the other hand, since the tree grows out from the center of the block, that is to say, the leaves occupy the perimeter; according to the "near" criterion, the leaves should be symmetric.

Based on the above analysis, the following constraints are listed:

- The tree grows out from the center, and the perimeter consists of leaves;
- One tree will be a root which has no offspring;
- Except for the root and the leaves, each node has four child nodes;
- Every parent node and its four child nodes should have at least 1 pixel adjacent;
- The leaves must be symmetric in both vertical and horizontal directions;
- If a parent node's four child nodes are leaves, the four child nodes must be adjacent pixels; otherwise, adjacent pixels are, at least, preferable.

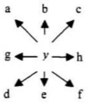


Figure 4.4 Adjacent pixels.

The adjacent pixel is defined in this way: if and only if a pixel $x$ is in one of the eight positions, which are a, b, c, d, e, f, g, and h, as shown in Figure 4.4, the pixel $x$ is considered as the adjacent pixel of pixel $y$. Pixel $x$ and $y$ are called adjacent pixels.

### 4.2.2 Possible 8 × 8 Patterns

Complying with the constraints described above, patterns are constructed from the perimeter to the core. The procedures are described below.

According to the constraints, if four pixels are leaves, they should be adjacent pixels. Some possible connections (shapes) of four pixels are shown in Figure 4.5. All these shapes as well as their rotated formats satisfy such constraints. In shape (a), any

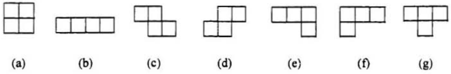


Figure 4.5 Examples of four connected pixels.

two of the four pixels are adjacent pixels. Shape (b) has a line structure thus it can represent more horizontal information than other shapes. Although shapes (c), (d), (e), (f),

(g) and their rotated formats meet the constraints, through exhaustive simulations, it has been found if the leaves have these shapes, it is impossible to arrange the other pixels according to the constraints. Hence, the possible shapes of the leaves are (a) and (b).

With (a) and (b), there are five possible kinds of leaf shape combinations at the top and the bottom of a block: (a)(a)(a)(a), (b)(b), (a)(b)(a), (b)(a)(a), and (a)(a)(b), which are illustrated in Figure 4.6. In these combinations, (D) and (E) do not conform to

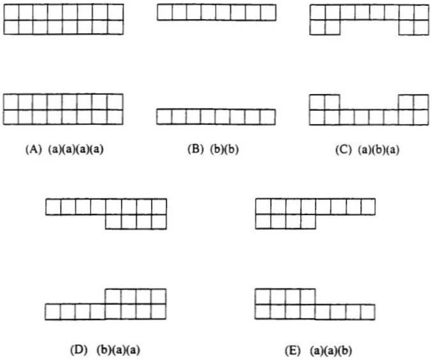


Figure 4.6 Possible kinds of leaf shape combinations.

the constraints since the leaves formed with the two basic shapes are not symmetric in both the horizontal and vertical directions, so they should be abandoned. By now, three kinds of qualified leaf shape combinations at the top and the bottom of a block are obtained, which are (A), (B), and (C) in Figure4.6.

Based on (A), pattern #1 (shown in Figure 4.7) is constructed. In pattern #1, one of the pixels labeled with a star(*) in the centric part of the block is the root (star root) that has no child nodes, and the other three pixels labeled with 5d, 5e, and 4e are tree roots that have four child nodes each. The star root (4d) together with the three tree roots

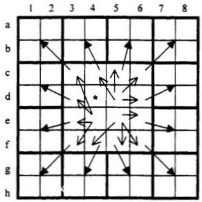
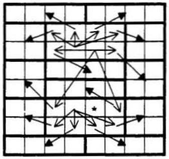


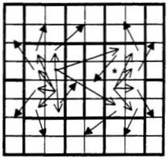
Figure 4.7 Pattern #1.

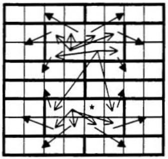
(5d, 5e, and 4e) form the root area of the pattern. The first level of the pattern consists of pixels labeled with 3c, 4c, 5c, 6c, 3d, 3e, 3f, 4f, 5f, 6f, 6e, and 6d. Each node in the first level also has four pixels as its children, which are tree leaves and form the second level. Therefore, except for the star root, the other pixels in an 8 × 8 pattern are connected in three quad-trees.

By changing the position of the star root and the relationship between the roots and their child nodes in the first level in pattern #1, sixteen patterns (including pattern #1) are generated.

Similarly, based on (B), forty possible patterns are obtained. Figure 4.8 illustrates some of them. Note pattern #3, #5, and #7 are the vertical formats of pattern #2, #4, and #6 respectively.


Pattern #2


Pattern #3


Pattern #4

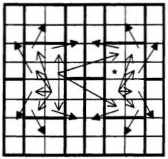
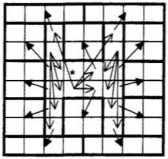

Pattern #5

Figure 4.8 Some 8 × 8 patterns based on Figure 4.6 (B).


Pattern #6

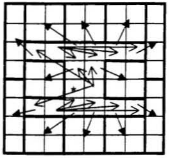
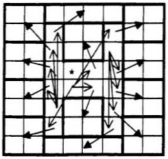

Pattern #7

Figure 4.8 Some 8 × 8 patterns based on Figure 4.6 (B). (Continued)

Also, based on (C), sixteen patterns are built. Figure 4.9 shows two of them, where pattern #9 is the vertical format of pattern #8.


Pattern #8

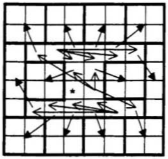

Pattern #9

Figure 4.9 Some 8 × 8 patterns based on Figure 4.6 (C).

In a nutshell, with the three basic models (Figure 4.6 (A), (B), (C)), seventy-two 8 × 8 patterns are constructed in total, which satisfy all of the constraints.

### 4.2.3 Ordering Residue Picture with 8 × 8 Patterns

After 8 × 8 patterns are determined, the problem is how to replace the DWT with 8 × 8 patterns in zerotree residue coding. To clearly explain this problem, an example is given below.

Suppose the resolution of a residue picture is 16 × 16 and pattern #1 (Figure 4.7) is chosen. What needs to be done is:

1) Separate the residue picture into blocks of 8 × 8. That is to say, four 8 × 8 blocks are obtained.

2) Apply pattern #1 to each block. Refer to Figure 4.10. Then, each pixel in a block is considered as a node in a tree. In each block, the star pixel is the star root that has no offspring, and the other pixels have three quad-tree style connections.

3) Take all the roots (including all the star roots) from each block, and then put them into the first 4 × 4 areas of the ordered picture as shown in Figure 4.11. So roots labeled with d4, 5d, 4e, and 5e in the first block occupy the first 2 × 2 areas of the ordered picture, roots labeled with d12, 13d, 12e, and 13e in the second block are with the second 2 × 2 positions, and so on.

4) Take all the first level child nodes from each block, which are labeled as 4c, 5c, 6c, 6d, 3c, 3d, 3e, 3f, 6e, 6f, 5f, 4f, etc. Following the occupied 4 × 4 areas, as illustrated in Figure 4.11, insert them into the next upper-right, lower-left, and lower-right areas.

5) Take all the second level child nodes from each block, which are also leaves of quad-trees. As in step 4), arrange them into successive areas of the ordered picture (Figure 4.11).

Finally, the ordered picture (Figure 4.11) is composed. It has a tree structure such that

- Each node, except for the star roots and those in the last level (the leaves), has four child nodes;
- A node, combined with all its descendants, forms a spatial orientation tree.

Now, zerotree coding can be applied to the pixels in the ordered residue picture.

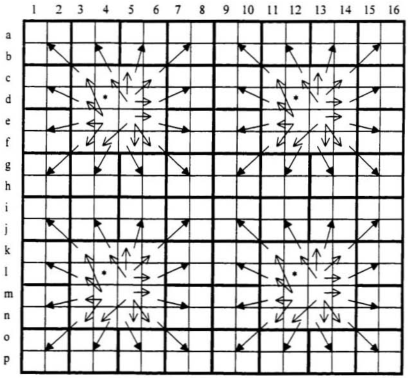
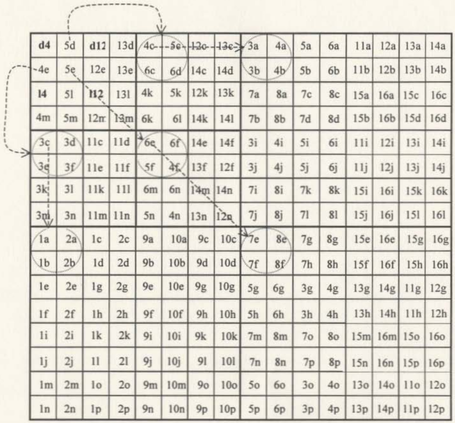


Figure 4.10 Applying pattern #1 to a 16 × 16 picture.



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| d4 | 5d | d12 | 13d | 4c | 5c | 12c | 13c | 3a | 4a | 5a | 6a | 11a | 12a | 13a | 14a |
| 4e | 5e | 12e | 13e | 6c | 6d | 14c | 14d | 3b | 4b | 5b | 6b | 11b | 12b | 13b | 14b |
| l4 | 5l | l12 | 13l | 4k | 5k | 12k | 13k | 7a | 8a | 7c | 8c | 15a | 16a | 15c | 16c |
| 4m | 5m | 12m | 13m | 6k | 6l | 14k | 14l | 7b | 8b | 7d | 8d | 15b | 16b | 15d | 16d |
| 3c | 3d | 11c | 11d | 6e | 6f | 14e | 14f | 3i | 4i | 5i | 6i | 11i | 12i | 13i | 14i |
| 3e | 3f | 11e | 11f | 5f | 4f | 13f | 12f | 3j | 4j | 5j | 6j | 11j | 12j | 13j | 14j |
| 3k | 3l | 11k | 11l | 6m | 6n | 14m | 14n | 7i | 8i | 7k | 8k | 15i | 16i | 15k | 16k |
| 3m | 3n | 11m | 11n | 5n | 4n | 13n | 12n | 7j | 8j | 7l | 8l | 15j | 16j | 15l | 16l |
| 1a | 2a | 1c | 2c | 9a | 10a | 9c | 10c | 7e | 8e | 7g | 8g | 15e | 16e | 15g | 16g |
| 1b | 2b | 1d | 2d | 9b | 10b | 9d | 10d | 7f | 8f | 7h | 8h | 15f | 16f | 15h | 16h |
| 1e | 2e | 1g | 2g | 9e | 10e | 9g | 10g | 5g | 6g | 3g | 4g | 13g | 14g | 11g | 12g |
| 1f | 2f | 1h | 2h | 9f | 10f | 9h | 10h | 5h | 6h | 3h | 4h | 13h | 14h | 11h | 12h |
| 1i | 2i | 1k | 2k | 9i | 10i | 9k | 10k | 7m | 8m | 7o | 8o | 15m | 16m | 15o | 16o |
| 1j | 2j | 1l | 2l | 9j | 10j | 9l | 10l | 7n | 8n | 7p | 8p | 15n | 16n | 15p | 16p |
| 1m | 2m | 1o | 2o | 9m | 10m | 9o | 10o | 5o | 6o | 3o | 4o | 13o | 14o | 11o | 12o |
| 1n | 2n | 1p | 2p | 9n | 10n | 9p | 10p | 5p | 6p | 3p | 4p | 13p | 14p | 11p | 12p |

Figure 4.11 Ordered 16 × 16 picture with pattern #1.

## 4.3 16 $\times$ 16 Ordering Patterns

Besides 8 $\times$ 8 patterns, some 16 $\times$ 16 patterns are constructed. Similar to the 8 $\times$ 8 case, a set of constraints are also needed for building 16 $\times$ 16 patterns.

### 4.3.1 Constraints for Building 16 $\times$ 16 patterns

16 $\times$ 16 patterns are extensions of 8 $\times$ 8 patterns. Refering to the analysis in section 4.2.1, to arrange an 8 $\times$ 8 block pixels in quad-trees, analysis can be done as before:

$$am + b = 2^n \times 2^n \tag{4.5}$$

For a 16 $\times$ 16 block ($n = 4$), equation (4.5) is rewritten as

$$am + b = 256. \tag{4.6}$$

If $b = 0$ (no pixel left case), equation (4.6) is untenable. If $b = 1$ (1 pixel left case), the solutions of equation (4.6) are $m = 5$, $a = 51$ or $m = 85$, $a = 3$. That means the 255 pixels can be arranged in fifty-one 1-level quad-trees or three 3-level quad-trees. By modifying the constraints for building 8 $\times$ 8 patterns, the constraints for building 16 $\times$ 16 patterns are obtained. Most of them are the same as the former, such as the tree leaves must be adjacent pixels and symmetric. The difference is that the parent nodes of the leaves are not required to be adjacent with the leaves although 1 pixel adjacent is preferable.

Here are the constraints for building 16 $\times$ 16 patterns:

- The tree grows out from the center, and the perimeter consists of leaves;
- One tree will be a root which has no offspring;
- Except for the root and the leaves, each node has four child nodes;
- Except for the parent nodes whose child nodes are leaves, every other parent node and it's four child nodes must be at least 1 pixel adjacent;

- The leaves must be symmetric in both horizontal and vertical directions:
- If a parent node's four child nodes are leaves, the four child nodes must be adjacent pixels; otherwise, adjacent pixels are, at least, preferable.

### 4.3.2 Examples of 16 × 16 Patterns

Following the constraints described above, some 16 × 16 patterns are built. Most of the 16 × 16 patterns are the extension of 8 × 8 patterns.

Figure 4.12 – 4.14 shows three examples of 16 × 16 patterns.

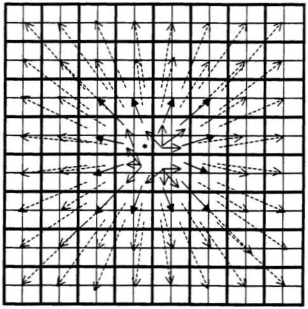


Figure 4.12 Pattern #10

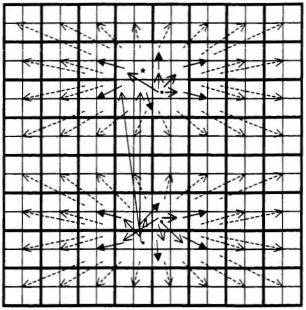


Figure 4.13 Pattern #11

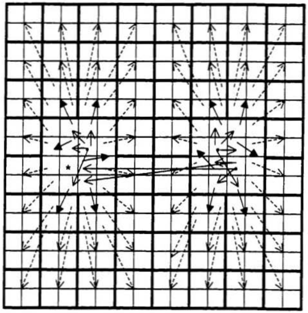


Figure 4.14 Pattern #12

## 4.3.3 Ordering Residue Picture with 16 × 16 pattern

The method of ordering the residue picture with 16 × 16 patterns is similar to that with 8 × 8 patterns. First, the residue picture is separated into 16 × 16 blocks. And then, a 16 × 16 pattern is applied to each block. Next the roots, the first level child nodes, the second level child nodes, and the third level child nodes are moved to certain positions in the ordered picture.

To simplify the explanation, suppose the size of the residue picture is 16 × 16, so it does not need to be further segmented, and pattern #10 is used for ordering pixels in the residue picture. After changing positions of pixels, the ordered picture is shown in Figure 4.15.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| h8 | 9h | 8g | 9g | 7e | 8e | 9e | 10e | 5a | 6a | 7a | 8a | 9a | 10a | 11a | 12a |
| 8i | 9i | 10g | 10h | 7f | 8f | 9f | 10f | 5b | 6b | 7b | 8b | 9b | 10b | 11b | 12b |
| 7g | 7h | 10i | 10j | 11e | 12e | 11g | 12g | 5c | 6c | 7c | 8c | 9c | 10c | 11c | 12c |
| 7i | 7j | 9j | 8j | 11f | 12f | 11h | 12h | 5d | 6d | 7d | 8d | 9d | 10d | 11d | 12d |
| 5e | 6e | 5g | 6g | 11i | 12i | 11k | 12k | 13a | 14a | 15a | 16a | 13e | 14e | 15e | 16e |
| 5f | 6f | 5h | 6h | 11j | 12j | 11l | 12l | 13b | 14b | 15b | 16b | 13f | 14f | 15f | 16f |
| 5i | 6i | 5k | 6k | 9k | 10k | 7k | 8k | 13c | 14c | 15c | 16c | 13g | 14g | 15g | 16g |
| 5j | 6j | 5l | 6l | 9l | 10l | 7l | 8l | 13d | 14d | 15d | 16d | 13h | 14h | 15h | 16h |
| 1a | 2a | 3a | 4a | 1e | 2e | 3e | 4e | 13i | 14i | 15i | 16i | 13m | 14m | 15m | 16m |
| 1b | 2b | 3b | 4b | 1f | 2f | 3f | 4f | 13j | 14j | 15j | 16j | 13n | 14n | 15n | 16n |
| 1c | 2c | 3c | 4c | 1g | 2g | 3g | 4g | 13k | 14k | 15k | 16k | 13o | 14o | 15o | 16o |
| 1d | 2d | 3d | 4d | 1h | 2h | 3h | 4h | 13l | 14l | 15l | 16l | 13p | 14p | 15p | 16p |
| 1i | 2i | 3i | 4i | 1m | 2m | 3m | 4m | 9m | 10m | 11m | 12m | 5m | 6m | 7m | 8m |
| 1j | 2j | 3j | 4j | 1n | 2n | 3n | 4n | 9n | 10n | 11n | 12n | 5n | 6n | 7n | 8n |
| 1k | 2k | 3k | 4k | 1o | 2o | 3o | 4o | 9o | 10o | 11o | 12o | 5o | 6o | 7o | 8o |
| 1l | 2l | 3l | 4l | 1p | 2p | 3p | 4p | 9p | 10p | 11p | 12p | 5p | 6p | 7p | 8p |

Figure 4.15 Ordered 16 × 16 picture with pattern #10.

## 4.4 Arbitrary Level Patterns and Orientation Patterns

Arbitrary level patterns and orientation patterns evolve from the patterns developed.

### 4.4.1 Arbitrary Level Patterns

The 8 × 8 patterns are two level patterns and the 16 × 16 patterns are three level patterns. Based on pattern #1 (8 × 8) and pattern #10 (16 × 16), this kind of pattern can be expanded to arbitrary level, such as a four level pattern (32 × 32) (Figure 4.16), a five

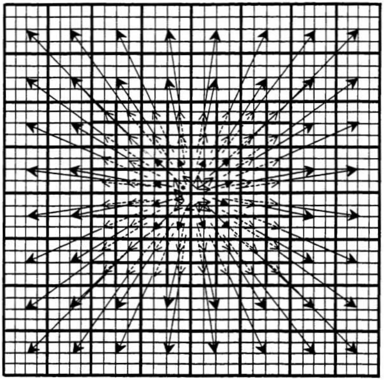


Figure 4.16 Pattern #13

level pattern (64 × 64), and a six level pattern (128 × 128). The ordering method resembles what has been introduced in section 4.2.3 and 4.3.3.

### 4.4.2 Orientated Patterns

Because there is often a lot of horizontal and vertical motion in an image, the residue picture contains horizontal and vertical blocks of pixels with zero gray-values. To efficiently concatenate these pixels with zerotrees, orientation pattern pairs are applied to blocks of a segmented residue picture.

The orientation pattern pair consists of two patterns: one is a horizontal pattern, the other is a vertical pattern. For instance, pattern #2 and pattern #3 (Figure 4.8) is a pair of orientation patterns. The same applies to pattern #4 and #5, pattern #6 and #7 (Figure 4.8), pattern #8 and #9 (Figure 4.9), as well as pattern #11 (Figure 4.13) and #12 (Figure 4.14).

However, the process of applying orientation patterns to a residue picture is a little different from the usual way described in section 4.2.3 and 4.3.3. After segmenting a residue picture into blocks (such as 8 × 8 blocks), some criteria are defined for using orientation patterns.

Here are the criteria:

- calculate activity for each column of a block,
- calculate activity for each row of a block,
- choose the direction (horizontal or vertical) which minimizes the amount of activity.

The activity is defined as the frequency of the occurrence of nonzero-gray values in a column or a row of a block. Note that when the block is coded, an extra bit has to be set to indicate which of two patterns is being used.

## 4.5 Zerotree Coding of Ordered Residue Picture

With the help of the pattern, residue pictures are ordered. The ordered picture has the same hierarchical tree structure as shown in Figure 2.11, where the spatial relationship of pixels is defined. The zerotree coding algorithm performed on the ordered residue pictures in the proposed scheme is the Set Partitioning In Hierarchical Trees (SPIHT) algorithm, which has been introduced in section 2.2.4. It is a progressive transmission scheme that incorporates two concepts: arranging the pixels by magnitude and transmitting the most significant bits first.

## 4.6 Summary

Due to the hundreds of thousands of possible ways to arrange pixels in a block, it is impossible to build patterns without any constraints and a recursive algorithm encompassing all patterns in all levels cannot be found. In this chapter, following the setting of some constraints, many different level patterns were designed. By using these patterns, prediction residue pictures were ordered; after that, zerotree coding was performed on these ordered residue pictures. Thus, a simple residue coding scheme, which is called zerotree pattern residue coding scheme, was developed.

The performances of the patterns will be reported in the next chapter as the elementary simulations.

# Chapter 5

# Pattern Selection Simulations

In this chapter, the patterns designed in chapter 4 are tested. The purpose is to find out the most suitable pattern for use in coding.

## 5.1 Simulation Procedures

The system diagram of the video coding scheme proposed in this thesis is shown in Figure 3.1. Different from the system simulation, the task of the pattern selection simulations is to test the effectiveness of patterns, so instead of image sequences, only two successive frames are used as input images.

Figure.5.1 is the simulation diagram. The simulations follow the following procedures.

Step 1: Select two successive frames at random from an image sequence.

Step 2: Segment the two frames into blocks of $16 \times 16$ size. By half-pixel block-matching method, obtain motion vectors for each block. (The threshold set for the motion estimation is 250. Refer to chapter 3.)

Step 3: Shift the blocks in the previous frame according to the motion vectors, then obtain the prediction picture of the current frame.

Step 4: Subtract the prediction picture from the current frame, thus obtaining the difference picture (residue).

Step 5: Code the difference picture by either the zerotree pattern coder, the zerotree wavelet coder, or the raw zerotree coder (Refer to section 4.2). The patterns used by the zerotree residue codec include $8 \times 8$ patterns, $16 \times 16$ patterns, and orientation patterns, which have been developed in chapter 4.

Step 6: Decode the compressed difference picture by the relevant decoder.

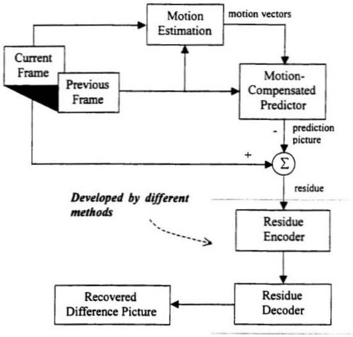


Figure 5.1 Simulation Diagram.

To evaluate the performance of the patterns, their application on various kinds of residue pictures is necessary. The difference picture is compressed at different compression rates. To measure the distortion of the residue coder (decoder), the peak signal to noise ratio (PSNR) is used.

The PSNR is defined as

$$PSNR = 10\log_{10}\left[\frac{255^2}{MSE}\right] \quad dB$$

where MSE denotes the mean squared error between the original and the reconstructed images. MSE is defined by

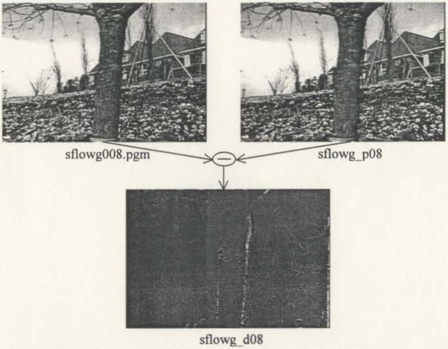
$$MSE = \sum_i \left[\frac{(p_i - q_i)^2}{N}\right]$$

where $p_i$ are the values of the reconstructed image, $q_i$ are the values of the original image, and $N$ is the size of both pictures.

## 5.2 Difference Pictures

Six frames are arbitrarily selected from three gray-scale image sequences: the seventh (sflowg007.pgm) and the eighth (sflowg008.pgm) frames of *Flower Garden* image sequence, the fourteenth (mo014.pgm) and the fifteenth (mo015.pgm) frames of *Mobile Calendar* image sequence, and the tenth (tt010.pgm) and the eleventh (tt011.pgm) frames of *Table Tennis* image sequence. The resolution of these frames is 352 × 240. Refer to section 6.1 for detailed information about these image sequences.

For each pair of frames mentioned above, the first four steps were applied (section 5.1) to obtain three difference pictures. These pictures have different characteristics. Figure 5.2 shows these difference pictures labeled with sflowg_d08, mo_d15, and tt_d11. For instance, sflowg_d08 is the difference picture between the eighth frame (sflowg008.pgm) and the prediction (sflowg_p08) for this frame. Similarly, mo_d15 is the residue between frame mo015.pgm and its prediction (mo_p15), and tt_d11 is the residue between frame tt011.pgm and its prediction tt_p11.



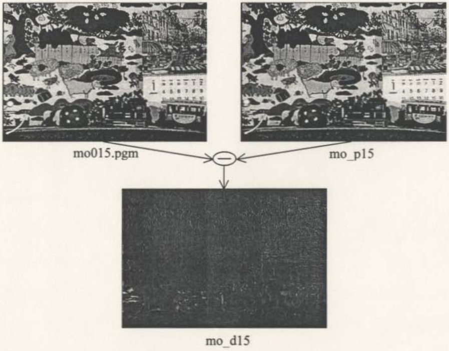


Figure 5.2 Test pictures.

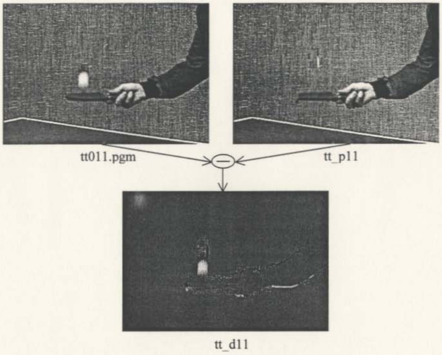


Figure 5.2 Test pictures. (Continued)

## 5.3 Results from Single Patterns

With the above test pictures, all the 8 × 8 patterns and 16 × 16 patterns developed in chapter 4 were tested, but only some representative simulation results are reported in the following tables.

Table 5.1 to 5.3 shows the peak signal to noise ratios (PSNR) of the recovered difference pictures at different compression rates with different residue coding methods. In these tables, the "Wavelet Transform" means the residues are coded with the zerotree wavelet coder. The "Raw" stands for raw zerotree coding – a strategy of coding residues with only zerotrees without applying DWT or patterns. The other methods, such as

"Pattern #1", "Pattern #2", etc., indicate the zerotree pattern residue coder with different patterns. Among the patterns, "Pattern #1" to "Pattern #9" are $8 \times 8$ patterns (2-level), "Pattern #10" to "Pattern #12" are $16 \times 16$ patterns (3-level), "Pattern #13" is $32 \times 32$

Table 5.1 Comparison of PSNR on sflowg_d08 (*Flower Garden*)

| Residue Coding Method | Compression Bit Rate | | |
|---|---|---|---|
| | bpp = 0.50 (bit/pixel) | bpp = 0.25 (bit/pixel) | bpp = 0.05 (bit/pixel) |
| Wavelet Transform | 35.98 | 33.73 | 30.94 |
| Raw | 36.07 | 33.62 | 30.78 |
| Pattern #1 (Figure 4.7) | 36.07 | 33.62 | 30.83 |
| Pattern #2 (Figure 4.8) | 35.96 | 33.56 | 30.74 |
| Pattern #3 (Figure 4.8) | 35.93 | 33.54 | 30.75 |
| Pattern #4 (Figure 4.8) | 35.97 | 33.56 | 30.73 |
| Pattern #5 (Figure 4.8) | 35.94 | 33.54 | 30.75 |
| Pattern #6 (Figure 4.8) | 35.98 | 33.55 | 30.77 |
| Pattern #7 (Figure 4.8) | 36.00 | 33.56 | 30.76 |
| Pattern #8 (Figure 4.9) | 35.92 | 33.55 | 30.76 |
| Pattern #9 (Figure 4.9) | 35.95 | 33.54 | 30.78 |
| Pattern #10 (Figure 4.12) | 36.15 | 33.74 | 30.85 |
| Pattern #11 (Figure 4.13) | 36.15 | 33.75 | 30.84 |
| Pattern #12 (Figure 4.14) | 36.13 | 33.72 | 30.91 |
| Pattern #13 (Figure 4.16) | 36.08 | 33.64 | 30.81 |
| Pattern-5level | 36.03 | 33.58 | 30.80 |
| Pattern-6level | 36.00 | 33.56 | 30.82 |
| Pattern-1level | 35.78 | 33.41 | 30.66 |

pattern (4-level). "Pattern-5level" and "Pattern-6level", which are the further extensions of "Pattern #13", are 64 × 64 (5-level) and 128 × 128 (6-level) patterns respectively. On the other hand, "Pattern-1level", which is the smallest version in arbitrary level pattern family (section 4.4.1), is a 4 × 4 pattern.

Table 5.2 Comparison of PSNR on mo_d15 (*Mobile Calendar*)

| Residue Coding Method | Compression Rate | | |
|---|---|---|---|
| | bpp = 0.50 (bit/pixel) | bpp = 0.20 (bit/pixel) | bpp = 0.03 (bit/pixel) |
| Wavelet Transform | 35.71 | 32.99 | 30.65 |
| Raw | 36.10 | 33.03 | 30.55 |
| Pattern #1 (Figure 4.7) | 36.13 | 33.04 | 30.63 |
| Pattern #2 (Figure 4.8) | 36.08 | 32.99 | 30.61 |
| Pattern #3 (Figure 4.8) | 36.06 | 32.95 | 30.60 |
| Pattern #4 (Figure 4.8) | 36.09 | 33.00 | 30.63 |
| Pattern #5 (Figure 4.8) | 36.06 | 32.95 | 30.59 |
| Pattern #6 (Figure 4.8) | 36.08 | 32.98 | 30.62 |
| Pattern #7 (Figure 4.8) | 36.07 | 32.98 | 30.62 |
| Pattern #8 (Figure 4.9 | 36.06 | 32.97 | 30.63 |
| Pattern #9 (Figure 4.9 | 36.06 | 32.97 | 30.62 |
| Pattern #10 (Figure 4.12) | 36.18 | 33.12 | 30.61 |
| Pattern #11 (Figure 4.13) | 36.20 | 33.14 | 30.62 |
| Pattern #12 (Figure 4.14) | 36.19 | 33.12 | 30.62 |
| Pattern #13 (Figure 4.16) | 36.15 | 33.08 | 30.60 |
| Pattern-5level | 36.08 | 33.03 | 30.56 |
| Pattern-6level | 36.09 | 33.01 | 30.52 |
| Pattern-1level | 35.94 | 32.87 | 30.56 |

Table 5.3 Comparison of PSNR on tt_d11 (*Table Tennis*)

| Residue Coding Method | Compression Rate | | |
|---|---|---|---|
| | bpp = 0.20 (bit/pixel) | bpp = 0.15 (bit/pixel) | bpp = 0.10 (bit/pixel) |
| Wavelet Transform | 38.26 | 37.72 | 36.68 |
| Raw | 38.35 | 37.57 | 36.15 |
| Pattern #1 (Figure 4.7 | 38.40 | 37.64 | 36.27 |
| Pattern #2 (Figure 4.8 | 38.35 | 37.59 | 36.21 |
| Pattern #3 (Figure 4.8 | 38.33 | 37.55 | 36.21 |
| Pattern #4 (Figure 4.8) | 38.33 | 37.56 | 36.19 |
| Pattern #5 (Figure 4.8) | 38.30 | 37.52 | 36.18 |
| Pattern #6 (Figure 4.8) | 38.31 | 37.54 | 36.18 |
| Pattern #7 (Figure 4.8) | 38.33 | 37.58 | 36.26 |
| Pattern #8 (Figure 4.9) | 38.32 | 37.55 | 36.17 |
| Pattern #9 (Figure 4.9) | 38.30 | 37.53 | 36.21 |
| Pattern #10 (Figure 4.12) | 38.47 | 37.84 | 36.40 |
| Pattern #11 (Figure 4.13) | 38.47 | 37.84 | 36.46 |
| Pattern #12 (Figure 4.14) | 38.49 | 37.82 | 36.34 |
| Pattern #13 (Figure 4.16) | 38.45 | 37.74 | 36.30 |
| Pattern-5level | 38.34 | 37.61 | 36.22 |
| Pattern-6level | 38.35 | 37.57 | 36.15 |
| Pattern-1level | 38.12 | 37.19 | 36.05 |

## 5.4 Results from Orientation Patterns

In this simulation, similar to section 5.3, the residue coding methods include zerotree wavelet coding algorithm ("Wavelet Transform" in the following tables), zerotree pattern coding algorithm (such as "Pattern #2 & Pattern #3" in the following tables), and raw zerotree coding strategy ("Raw" in the following tables). The difference is that the patterns used here are orientation pattern pairs.

For using orientation patterns, a flag (one bit) is needed for each block to indicate choice of pattern. So the cost of one bit per block should be included in the compressed data. For instance, in these simulations, the resolution of the pictures is 352 × 240, then an addition of 0.0156 bpp is used for the flags of 8 × 8 orientation pattern pairs, and 0.0039 bpp is used for the flags of 16 × 16 orientation pattern pairs (See Table 5.4 to 5.6). Table 5.4 to 5.5 illustrates the peak signal to noise ratios (PSNR) of the recovered

Table 5.4 Comparison of PSNR on sflowg_d08 (*Flower Garden*)

| Residue Coding Method | Compression Rate | | |
|---|---|---|---|
| | bpp = 0.50 + flags (bit/pixel) | Bpp = 0.25 + flags (bit/pixel) | bpp = 0.05 + flags (bit/pixel) |
| Wavelet Transform | 35.98 (+ 0) | 33.73 (+ 0) | 30.94 (+ 0) |
| Raw | 36.07 (+ 0) | 33.62 (+ 0) | 30.78 (+ 0) |
| Pattern #2 & Pattern #3 | 35.95 (+0.0156 bpp) | 33.57 (+0.0156 bpp) | 30.75 (+0.0156 bpp) |
| Pattern #4 & Pattern #5 | 36.00 (+0.0156 bpp) | 33.57 (+0.0156 bpp) | 30.74 (+0.0156 bpp) |
| Pattern #6 & Pattern #7 | 36.01 (+0.0156 bpp) | 33.58 (+0.0156 bpp) | 30.77 (+0.0156 bpp) |
| Pattern #8 & Pattern #9 | 35.95 (+0.0156 bpp) | 33.55 (+0.0156 bpp) | 30.79 (+0.0156 bpp) |
| Pattern #11 & Pattern #12 | 36.17 (+0.0039 bpp) | 33.76 (+0.0039 bpp) | 30.89 (+0.0039 bpp) |

difference pictures at different compression rates with different residue coding methods.

Table 5.5 Comparison of PSNR on mo_d15 (*Mobile Calendar*)

| Residue Coding Method | Compression Rate | | |
|---|---|---|---|
| | bpp = 0.50 + flags (bit/pixel) | bpp = 0.25 + flags (bit/pixel) | bpp = 0.03 - flags (bit/pixel) |
| Wavelet Transform | 35.71 (+ 0) | 32.99 (+ 0) | 30.65 (- 0) |
| Raw Zerotree Coding | 36.10 (+ 0) | 33.03 (+ 0) | 30.55 (- 0) |
| Pattern #2 & Pattern #3 | 36.08 (+0.0156 bpp) | 33.00 (+0.0156 bpp) | 30.61 (+0.0156 bpp) |
| Pattern #4 & Pattern #5 | 36.10 (+0.0156 bpp) | 32.95 (+0.0156 bpp) | 30.61 (+0.0156 bpp) |
| Pattern #6 & Pattern #7 | 36.09 (+0.0156 bpp) | 32.99 (+0.0156 bpp) | 30.61 (+0.0156 bpp) |
| Pattern #8 & Pattern #9 | 35.08 (+0.0156 bpp) | 32.97 (+0.0156 bpp) | 30.63 (+0.0156 bpp) |
| Pattern #11 & Pattern #12 | 36.22 (+0.0039 bpp) | 33.15 (+0.0039 bpp) | 30.63 (+0.0039 bpp) |

Table 5.6 Comparison of PSNR on tt_d11 (*Table Tennis*)

| Residue Coding Method | Compression Rate | | |
|---|---|---|---|
| | bpp = 0.20 + flags (bit/pixel) | bpp = 0.15 + flags (bit/pixel) | bpp = 0.10 + flags (bit/pixel) |
| Wavelet Transform | 38.26 (+ 0) | 37.72 (+ 0) | 36.68 (+ 0) |
| Raw Zerotree Coding | 38.35 (+ 0) | 37.57 (+ 0) | 36.15 (+ 0) |
| Pattern #2 & Pattern #3 | 38.34 (+0.0156 bpp) | 37.59 (+0.0156 bpp) | 36.21 (+0.0156 bpp) |
| Pattern #4 & Pattern #5 | 38.34 (+0.0156 bpp) | 37.56 (+0.0156 bpp) | 36.18 (+0.0156 bpp) |
| Pattern #6 & Pattern #7 | 38.33 (+0.0156 bpp) | 37.59 (+0.0156 bpp) | 36.24 (+0.0156 bpp) |
| Pattern #8 & Pattern #9 | 38.32 (+0.0156 bpp) | 37.56 (+0.0156 bpp) | 36.20 (+0.0156 bpp) |
| Pattern #11 & Pattern #12 | 38.51 (+0.0039 bpp) | 37.84 (+0.0039 bpp) | 36.46 (+0.0039 bpp) |

## 5.5 Comparison and Discussion

In view of the results obtained above, the performances of various patterns have little difference. Among them, the orientation pattern pair "Pattern #11 & Pattern #12" achieves higher PSNR than others. For convenience, from now on, this pattern pair is simply called "Pattern-best".

The high-level patterns, such as "Pattern-5level" and "Pattern-6level", cannot always work well because of the input image size. For example, if the image size is 352 × 240, it can not be separated into integer numbers of blocks of 32 × 32 (or 128 × 128); so there will be some areas that will not be ordered. That will influence the performance of such patterns. The lowest level pattern, "Pattern-1level" does not give good results.

"Pattern-best" seems to be the most suitable pattern for the zerotree pattern residue coder. Based on the results obtained in section 5.3, the effectiveness of using discrete wavelet transform zerotree coding ("DWT"), pattern zerotree coding ("Pattern-best"), or raw zerotree coding ("Raw") for dealing with residues is illustrated in Figure 5.3, 5.4, and 5.5.

Figure 5.3 to 5.5 show that the peak signal to noise ratios (PSNR) of the recovered difference pictures with "Pattern-best" are always higher than those with "Raw" method; and also higher than those with "DWT" at relative high bit rates, whereas a little lower at low bit rates. In these simulations, the difference pictures are coded at very low bit rates. However, in order to keep the reconstructed images in good quality (PSNR > 30 dB) in video coding, the residues cannot be compressed so much. Therefore, the advantage of the proposed zerotree pattern scheme for residue coding at relatively high bit rates is clear.

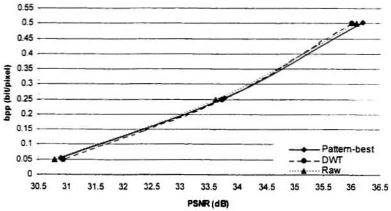


Figure 5.3 Bpp versus PSNR on sflowg_d08 (*Flower Garden*).

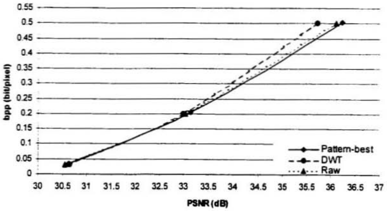


Figure 5.4 Bpp versus PSNR on mo_d15 (*Mobile Calendar*).

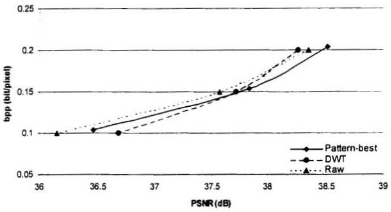


Figure 5.5 Bpp versus PSNR on tt_d11 (*Table Tennis*).

## 5.6 Summary

Comparative simulations on patterns were presented in this chapter. Three kinds of difference pictures were generated for the simulations. The simulation results were compared and discussed. "Pattern-best" is selected as the most suitable pattern on account of its good performance. Simulations for the recovery of entire image sequences is the topic of Chapter 6.

# Chapter 6

# Performance over Error Free Channels

The compression performance of the proposed video coding scheme over noiseless channels is reported in this chapter. The results are compared with those using the MPEG video compression standard and those with the zerotree wavelet transform residue coding method.

## 6.1 Selecting Image Sequences

For the simulations, four video sequences are chosen, which are *Flower Garden*, *Mobile Calendar*, *Table Tennis* (240 × 352, 8 bits/pixel, 25frames/s), and *Aircraft* (240 × 320, 8 bits/pixel, 30 frames/s). These images are sequences of gray-scale images. They contain brightness information only, no color information. The number of bits used for each pixel is 8 bits/pixel, which provide 256 (0-255) different brightness (gray) levels.

In detail, the image information of these sequences of frames is listed below.

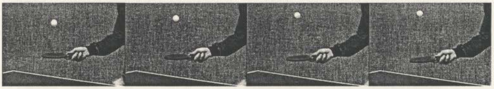
- *Flower Garden* is a trading shot of a garden with flowers: a tree in the foreground, and a house in the background; the house is partially covered by the tree. The frame rate of this sequence is 25 frames/s.
- *Mobile Calendar* shows a Toy Electrical Train moving around a room while a calendar moves vertically and the camera trades. The frame rate is 25 frames/s as well.
- *Table Tennis* shows one person bouncing a ping-pong ball. Only one arm and part of the body is visible. The frame rate is also 25 frames/s.

- *Aircraft* shows a graphical aircraft carrier crossing the picture from the right to the left and a text appears and disappears in the bottom. The frame rate is 30 frames/s.

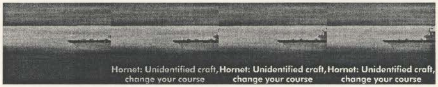
Some frames of the four sequences are illustrated in Figure 6.1.

(a) *Flower Garden*

(b) *Mobile Calendar*



(c) *Table Tennis*


Hornet: Unidentified craft, change your course
Hornet: Unidentified craft, change your course
Hornet: Unidentified craft, change your course

(d) *Aircraft*

Figure 6.1 Video sequences used in the simulations.

These sequences are interesting because of their complex motions and their different characteristics. Both *Mobile Calendar* and *Flower Garden* represent an object seen under a moving angle, where the displacement is not very great, and the latter contains very detailed areas. On the other hand, *Table Tennis* has very large motions, and there is a zoom in the middle of the sequence. Furthermore, *Aircraft* represents a typical example of a multimedia sequence mixing graphic and text.

## 6.2 Simulation Description

Following the system diagram (Figure 3.1), system simulations are conducted on the four sequences. The simulation procedure can be described as follows:

**Encoder:**

For frame 1 to frame $N$ in an image sequence, do

Step 1: Code the first frame (frame 1) by a wavelet technique, or by DCT as in MPEG, and send the coded file to the receiver.

Step 2: Set $n = 2$, frame ($n$-1) = recovered frame 1.

Step 3: Motion estimation: segment the frame ($n$-1) and the frame $n$ into $16 \times 16$ blocks, acquire motion vectors by the half-pixel accuracy block-matching technique. Refer to section 3.2.

Step 4: Motion-compensating predictor: obtain prediction picture ($P_n$) of frame $n$ by shifting the blocks in frame ($n$-1) in the light of the motion vectors. Refer to section 3.3.

Step 5: Obtain the residue (difference picture): residue = frame $n - P_n$.

Step 6: Code the residue by either the MPEG method (DCT), conventional zerotree wavelet coding, or zerotree pattern coding. Send the coded file to the receiver.

Step 7: Code the motion vectors (refer to section 3.3) and send the coded file to the receiver.

Step 8: Decode the residue using the relevant decoder, and then add it to the prediction picture $P_n$ to get the recovered picture $R_n$.

Step 9: Frame update: frame $(n-1) = R_n$.

Step 10: If $n < N$, $n = n + 1$, and then goto step 3.

**Receiver:**

Step 1: Get the compressed frame 1 file, decode it, then reconstruct the first frame.

Step 2: Set $n = 1$.

Step 3: Get the coded motion vectors and the coded residue information. Decode them. Shift the blocks of the recovered frame n according to the motion vectors, and then add the residue to obtain the recovered frame $(n+1)$.

Step 4: If $n < N$, $n = n + 1$, goto step 3.

Every part of the encoder is implemented as introduced before. In the receiver, the decoder works much faster than the encoder since0 it does not have to do the block-matching.

The MPEG encoder (version 1.5, February 1, 1995, University of Berkeley) used for these simulations is a "two level" coder, which means that first a full-search block-matching is used with a area of 7 and afterwards, a local half-pixel accuracy match is carried out if required. Only the I-Frame (Intraframe) and P-Frame (Predicted frame) are used in MPEG for the simulations. In addition, the residues are coded with the DCT method as introduced in chapter 2, section 2.1.2.2.

Hence, three video coding schemes with similar motion estimation and motion compensation techniques but with different residue coding strategies (DCT, zerotree wavelet, and zerotree pattern) are tested in these simulations. Thus the performances of the three kinds of residue coding strategies can be evaluated.

Moreover, in the simulations, twenty-five frames are taken for each sequence. For the zerotree pattern coder, "Pattern-best" (section 5.4) is applied. Note that compressed data includes one bit per block to indicate choice of pattern. The average of the bitrate and PSNR, omitting the results of the first picture which is not obtained from a recovered frame, are given to show performance.

The bitrate is defined as

$$\text{Bitrate (Mbits/s)} = \text{bpp} \times \text{image size} \times \text{frequency of the frames (Hz)}.$$

where bpp is the bits per pixel.

## 6.3 Simulation Results

Complying with the above procedures, simulations are performed on the luminance component of the four video sequences: *Flower Garden*, *Mobile Calendar*, *Table Tennis*, and *Aircraft*. The simulations assume that there is no noise in the transmission channel.

### 6.3.1 Compression Performance on *Flower Garden* Video

The peak signal to noise ratios (PSNR) measured in dB for each recovered frame at a constant bitrate is listed in Table 6.1, and the average PSNR for the 24 frames is computed as well. Figure 6.2 shows the curves obtained on various average bitrates and average peak signal to noise ratios. In the table and the figure, the item "1/2pel +

Table 6.1 Comparison of PSNR on *Flower Garden*

| Frame number | MPEG (Bitrate =2.4 Mbits/s) | 1/2pel + ZT_DWT (Bitrate = 2.15 Mbits/s) | 1/2pel + ZT_Pattern (Bitrate = 2.16 Mbits/s) |
|---|---|---|---|
| | Recovered (dB) | Recovered (dB) | Recovered (dB) |
| 2 | 33.44 | 36.68 | 36.62 |
| 3 | 33.23 | 34.89 | 34.66 |
| 4 | 33.11 | 34.71 | 34.71 |
| 5 | 32.96 | 33.68 | 32.98 |
| 6 | 32.97 | 32.85 | 32.01 |
| 7 | 32.83 | 33.81 | 33.09 |
| 8 | 32.69 | 33.75 | 33.24 |
| 9 | 32.77 | 34.10 | 33.71 |
| 10 | 32.59 | 33.88 | 33.49 |
| 11 | 32.63 | 32.59 | 31.96 |
| 12 | 32.63 | 31.66 | 30.99 |
| 13 | 32.43 | 31.25 | 30.61 |
| 14 | 32.31 | 32.83 | 32.25 |
| 15 | 32.42 | 34.13 | 33.23 |
| 16 | 32.31 | 33.33 | 33.00 |
| 17 | 32.19 | 33.21 | 32.69 |
| 18 | 31.62 | 31.34 | 30.65 |
| 19 | 31.69 | 31.09 | 30.40 |
| 20 | 31.63 | 32.38 | 31.84 |
| 21 | 31.62 | 32.87 | 32.27 |
| 22 | 31.68 | 33.16 | 32.63 |
| 23 | 31.73 | 32.85 | 32.47 |
| 24 | 31.84 | 33.66 | 33.24 |
| 25 | 31.86 | 32.71 | 32.14 |
| Average | 32.38 | 33.23 | 32.70 |

ZT_DWT" stands for half-pixel accuracy block-matching technique for motion estimation and motion compensation and zerotree wavelet coding for residues. Similarly, the item "1/2pel + ZT_Pattern" means that the same motion estimation and motion compensation technique is performed but the residues are coded by the zerotree pattern coder (with "Pattern-best"). More discussions of the issues around the comparison of the results are given in section 6.4.

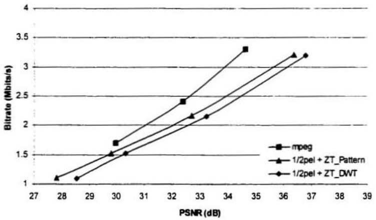


Figure 6.2 Average bitrate versus average PSNR on *Flower Garden*.

## 6.3.2 Compression Performance on *Table Tennis* Video

Similar to section 6.3.1, the performances of DCT, zerotree wavelet, and zerotree pattern residue coding methods on *Table Tennis* are shown in Figure 6.3.

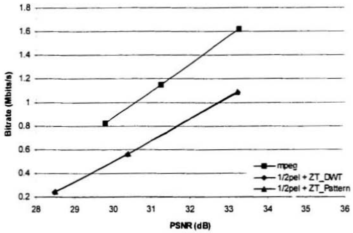


Figure 6.3 Average bitrate versus average PSNR on *Table Tennis*.

### 6.3.3 Compression Performance on *Mobile Calendar* Video

Similar to section 6.3.1, the performances of DCT, zerotree wavelet, and zerotree pattern residue coding methods on image sequence *Mobile Calendar* are shown in Table 6.2 and Figure 6.4.

### 6.3.4 Compression Performance on *Aircraft* Video

The same as in section 6.3.1, the performances of DCT, zerotree wavelet, and zerotree pattern residue coding methods on image sequence *Aircraft* are shown in Figure 6.5.

Table 6.2 Comparison of PSNR on *Mobile Calendar*

| Frame Number | MPEG (Bitrate = 2.2 Mbits/s) | 1/2pel + ZT_DWT (Bitrate = 2.14 Mbits/s) | 1/2pel + ZT_Pattern (Bitrate = 2.15 Mbits/s) |
|---|---|---|---|
| | Recovered (dB) | Recovered (dB) | Recovered (dB) |
| 2 | 29.48 | 32.38 | 33.56 |
| 3 | 28.92 | 29.93 | 30.56 |
| 4 | 28.56 | 29.76 | 30.55 |
| 5 | 28.14 | 29.23 | 30.06 |
| 6 | 27.75 | 28.99 | 29.85 |
| 7 | 27.69 | 28.95 | 29.71 |
| 8 | 27.19 | 28.72 | 29.39 |
| 9 | 27.31 | 29.18 | 29.91 |
| 10 | 27.63 | 29.39 | 30.04 |
| 11 | 28.41 | 29.69 | 30.37 |
| 12 | 28.71 | 30.06 | 30.76 |
| 13 | 28.96 | 30.32 | 31.36 |
| 14 | 28.95 | 30.53 | 31.57 |
| 15 | 28.98 | 30.41 | 31.50 |
| 16 | 29.16 | 30.10 | 31.09 |
| 17 | 29.22 | 30.20 | 31.09 |
| 18 | 29.15 | 30.18 | 31.14 |
| 19 | 29.25 | 30.41 | 31.08 |
| 20 | 29.22 | 30.22 | 30.89 |
| 21 | 29.28 | 30.35 | 30.99 |
| 22 | 29.31 | 30.02 | 30.95 |
| 23 | 29.33 | 30.21 | 30.93 |
| 24 | 29.32 | 30.26 | 30.95 |
| 25 | 29.35 | 30.57 | 31.17 |
| Average | 28.71 | 30.00 | 30.81 |

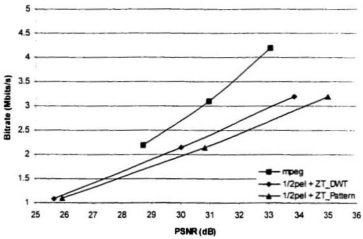


Figure 6.4 Average bitrate versus average PSNR on *Mobile Calendar*.

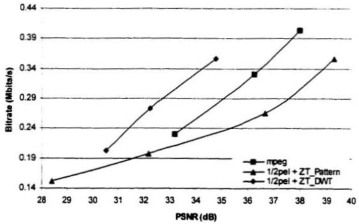


Figure 6.5 Average bitrate versus average PSNR on *Aircraft*.

### 6.3.5 Testing the Effect of Entropy Coding on Zerotree Coding

In all of the above simulations, for both the conventional zerotree wavelet coding scheme and the proposed zerotree pattern coding scheme, entropy coding (arithmetic coding) is applied to improve the efficiency of the algorithms. To test the influence of entropy coding on the two zerotree coding schemes, one more simulation is done on *Mobile Calendar* sequence. In this simulation, the arithmetic coding part in both of the zerotree wavelet coder and the zerotree pattern coder is deleted.

Figure 6.6 shows the comparison result. In Figure 6.6, "1/2pel + ZT_DWT_fast" and "1/2pel + ZT_Pattern_fast" stand for the results of the cases without entropy coding.

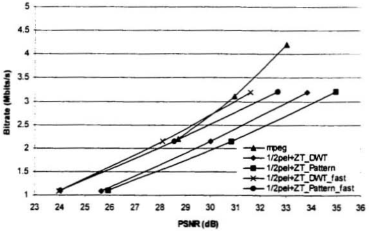


Figure 6.6 Average bitrate versus average PSNR on *Mobile Calendar* where the results of the zerotree wavelet coding and the zerotree pattern coding without entropy coding are included.

## 6.4 Discussion

Residue coding is one part of a video coding scheme. Its performance will influence the effect of the entire video coding scheme. The purpose of the simulations in this chapter is to test the proposed residue coding strategy, which is zerotree pattern coding method, and compare its results with those using MPEG method (DCT) residue coding or zerotree wavelet coding for residues. To evaluate the performances of the three residue coding strategies, the other part of a video coding scheme (motion estimation and motion compensation) is designed with the same algorithm (half-pixel accuracy block-matching), and motion vectors are coded with the same Huffman coding method. Therefore, the compression results of video coding reflect the performances of residue coding methods.

With the conventional zerotree wavelet coding scheme for residue coding, the simulation results on the first three different kinds of image sequences show that the use of zerotree wavelet coding scheme is more efficient than the DCT method in MPEG. For instance, on *Flower Garden*, the PSNR is improved 1.50 dB on an initial PSNR of 29.95 dB for a constant bitrate 1.70 Mbits/s. In terms of bitrate, the improvement is 16.47% (Figure 6.2).

Comparing the performance of two zerotree coding schemes - zerotree wavelet coding (with transforms) and zerotree pattern coding (without transforms) - the simulations show the following results.

For the *Flower Garden* sequence, which contains lots of complicated detail, the zerotree wavelet coder outperforms the zerotree pattern coder by about 0.50 dB on average (Figure 6.2). With the *Table Tennis* sequence, the results illustrate that the two methods have almost the same effect (Figure 6.3). However, considering the *Mobile*

*Calendar* sequence, which contains lots of motion, the zerotree pattern coder outperforms the zerotree wavelet coder by about 0.81 dB for a constant bitrate 2.14 Mbits/s (Figure 6.4). Moreover, for the 3-D animation *Aircraft* sequence, which consists of very complicated structures and textures, our zerotree pattern method also significantly outperforms the zerotree wavelet method (Figure 6.5).

Furthermore, as shown in Figure 6.6, the proposed residue coding scheme also achieves improvement when entropy coding is not used.

For computational comparison of the zerotree wavelet strategy and the zerotree pattern strategy,

In general, the results illustrate that zerotree coding schemes yield better performances than the DCT method in MPEG on average, and the zerotree pattern coding scheme without spatial transforms allows about the same compression as the zerotree wavelet coding method. On the other hand, the results show that the high compression performance of zerotree coding is mainly due to the zerotree apporach.

## 6.5 Summary

This chapter introduces the simulations of video coding over noiseless channels. The aim is to test the proposed residue coding scheme (zerotree pattern coding). Four kinds of video sequences were selected because of their various characteristics. Simulation results demonstrate that zerotree pattern coding performs close to the state-of-the-art technique in error free channels. The simulations for channels with errors will be reported in chapter 7.

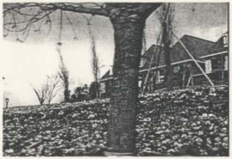
# Chapter 7

# Channel Error Resilience and Concealment

The problem of designing video coders that give good performance over channels with unpredictable bit error rates (BER), while maintaining the compression achievable with standard coding schemes designed for error free channels is very important in video transmission over cellular networks, weak radio links, or other noisy communication lines. In this chapter, the channel error resilience and concealment of the video coding scheme proposed in the thesis is examined by first conducting simulations on two frames of video sequences and then on entire video sequences.

## 7.1 Simulations on Two Frames of Video Sequences

In this section, simulations are performed on two frames from each video sequence of *Flower Garden* and *Football*. Figure 7.1 shows these test frames.


(a) Frame 7 of *Flower Garden*

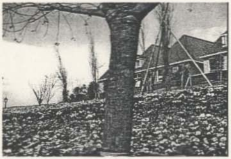
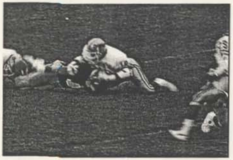
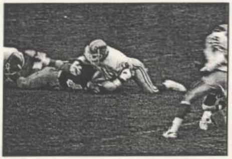

(b) Frame 8 of *Flower Garden*

Figure 7.1 Test frames from *Flower Garden* and *Football*.



(c) Frame 1 of *Football*



(d) Frame 2 of *Football*

Figure 7.1 Test frames from *Flower Garden* and *Football*. (continued)

There is no specific requirement for the choice of frames. The simulation results are given in section 7.2.

## 7.1.1 Simulation Description

In chapter 6, the simulations were conducted over noiseless channels; that is to say, the coded file is transmitted to the receiver without any errors. However, in this chapter, transmission of the coded file is simulated over a noisy channel.

The simulation procedures are similar to those described in chapter 6 (section 6.2). The important remarks on the simulations are described below.

- Suppose the first frame (frame 7 of *Flower Garden* and frame 1 of *Football*) is transmitted without suffering errors.
- Estimate motion and apply motion compensation as in section 6.2.
- Code residues either by a zerotree wavelet coder or a zerotree pattern coder. For each residue coder, there are two options: with entropy coding or without entropy coding. Note that "Pattern-best" (section 5.5) is applied for zerotree pattern coding.

- Add random errors into the coded residue bit stream. Random errors are added by a program which generates random bit errors at a pre-specified average rate called the bit error rate. For instance, if the bit error rate is 0.1, that means that on average it inverts one in every ten bits. In the simulations, random errors are added at a bit error rate (BER) of 0.1, 0.02, 0.01, or 0.001.
- Protect the coded motion vectors and the first 7 bytes (the header) of the coded residue stream from suffering errors.
- In these simulations, the additional 0.0039 bpp (section 5.4) for flags of patterns is omitted. The testing data rate is 0.53 bpp, 0.23 bpp, 0.13 bpp, or 0.08 bpp respectively (overall data rates, including motion vectors and the residue but not including the forward error correction overhead for the motion vectors and header), but only the results of **0.13 bpp** are presented in section 7.2. The corresponding results at an overall data rate of 0.53 bpp, 0.23 bpp, or 0.08 bpp are attached in *appendix* I, II, and III.
- Since the noise in the recovered images with patterns has an impulse (salt-and-pepper) distribution, and median filters work quite well in this context, so a median filter is applied to the recovery of patterns as simple post-processing. The size of the local neighborhood in the median filter is $3 \times 3$.

### 7.1.2 Performance of Raw Zerotree Coding over Noisy Channels

In the simulation description (section 7.1.1), it has been explained that residues are coded by zerotree coding with wavelets or ordering patterns in the following channel error resilience and concealment simulations. However, to measure the performance of raw zerotree coding (which means zerotrees without wavelets or patterns) on channel errors, some additional simulations are conducted in this section. Here the simulation procedures are similar to those introduced in section 7.1.1 except that residues are coded by a raw zerotree coder with no wavelet transforms or patterns.

Figure 7.2 shows the simulation results at an overall bit rate of 0.53 bpp. On visual effect, due to the definition of the data structure, most of the noise in the recovered image accumulates in the upper-left part of the reconstructed frames so that the image information in this part is totally lost. Such loss can not be tolerated if the lost part contains important information. Therefore, this coding scheme can not be applied practically. The cause of such results is explained in section 7.3.3.

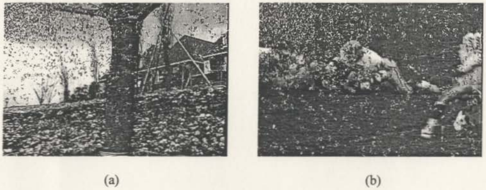

(a) (b)

Figure 7.2 Reconstructed frames from *Flower Garden* (a) and *Football* (b) where the residues are coded by a zerotree coder without wavelets and patterns, and coded residue information suffers errors (BER = 0.1).

## 7.2 Simulation Results (on two frames of video sequences)

Following the simulation procedures described in section 7.1.1, simulations are conducted on frames from *Flower Garden* and *Football*. Next are the simulation results at a data rate of 0.13 bpp.

### 7.2.1 For Frames from *Flower Garden* Video

Frame 7 and frame 8 (shown in Figure 7.1) of *Flower Garden* sequence are used in the simulations.

Figure 7.3 to 7.6 show the reconstructed frames where the coded residue information suffers errors. Table 7.1 summarizes these reconstructed frames and corresponding residue coding techniques.

Table 7.1 Summary of simulation results on reconstructed frames (*Flower Garden*)

| Bit error Rate | Entropy coding method | |
|---|---|---|
| | Arithmetic Coding | None |
| | ZT_DWT / ZT_Pattern | ZT_DWT_fast / ZT_Pattern_fast |
| BER = 0.1 | Figure 7.3 | Figure 7.5 |
| BER = 0.02 | | |
| BER = 0.01 | Figure 7.4 | Figure 7.6 |
| BER = 0.001 | | |

Figure 7.7 and 7.8 illustrate the average PSNR performance of video coding with different residue coding schemes on the virtual noisy channel. PSNR is the peak signal to noise ratio of the reconstructed frame. Simulations are repeated 70 times.

Figure 7.9 gives the post-processing results of applying a median filter on the reconstructed frames of Figure 7.3 (b), Figure 7.3 (d), Figure 7.4 (b), and Figure 7.4 (d).

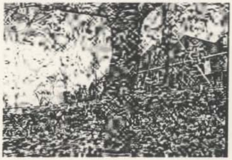
In the tables and the figures, the meaning of the abbreviations is explained as following:

with entropy coding (arithmetic coding):

- ZT_DWT — zerotree wavelet coding
- ZT_Pattern — zerotree pattern coding (Pattern-best)

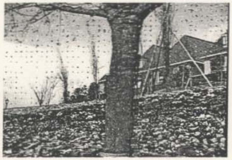
without entropy coding:

- ZT_DWT_fast — zerotree wavelet coding
- ZT_Pattern_fast — zerotree pattern coding (Pattern-best)



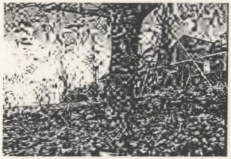
(a) ZT_DWT (BER = 0.1)

PSNR = 15.48 dB



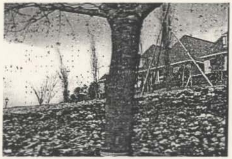
(b) ZT_Pattern (BER = 0.1)

PSNR = 19.78 dB



(c) ZT_DWT (BER = 0.02)

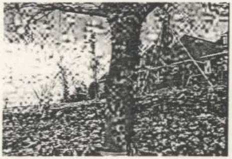
PSNR = 16.11 dB



(d) ZT_Pattern (BER = 0.02)

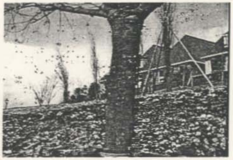
PSNR = 20.36 dB

Figure 7.3 Reconstructed frames from *Flower Garden* where the coded residue information suffers errors (BER = 0.1 or 0.02). Arithmetic coding is applied for residues.



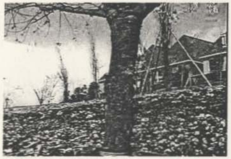
(a) ZT_DWT (BER = 0.01)

PSNR = 18.02 dB



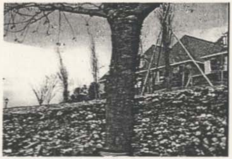
(b) ZT_Pattern (BER = 0.01)

PSNR = 21.54 dB



(c) ZT_DWT (BER = 0.001)

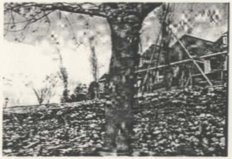
PSNR = 23.11 dB



(d) ZT_Pattern (BER = 0.001)

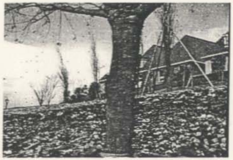
PSNR = 24.13 dB

Figure 7.4 Reconstructed frames from *Flower Garden* where the coded residue information suffers errors (BER = 0.01 or 0.001). Arithmetic coding is applied for residues.



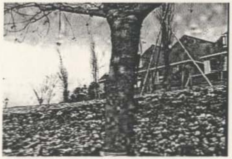
(a) ZT_DWT_fast (BER = 0.1)

PSNR = 18.05 dB



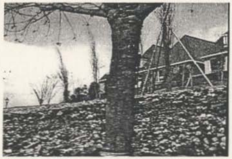
(b) ZT_Pattern_fast (BER = 0.1)

PSNR = 23.51 dB



(c) ZT_DWT_fast (BER = 0.02)

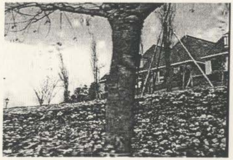
PSNR = 23.71 dB



(d) ZT_Pattern_fast (BER = 0.02)

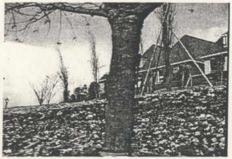
PSNR = 24.69 dB

Figure 7.5 Reconstructed frames from *Flower Garden* where the coded residue information suffers errors (BER = 0.1 or 0.02). No entropy coding for residues



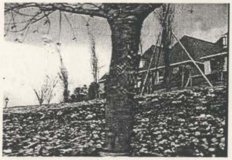
(a) ZT_DWT_fast (BER = 0.01)

PSNR = 24.09 dB



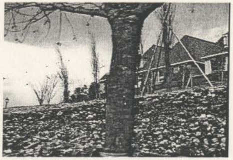
(b) ZT_Pattern_fast (BER = 0.01)

PSNR = 24.83 dB



(c) ZT_DWT_fast (BER = 0.001)

PSNR = 24.47 dB



(d) ZT_Pattern_fast (BER = 0.001)

PSNR = 24.98 dB

Figure 7.6 Reconstructed frames from *Flower Garden* where the coded residue information suffers errors (BER = 0.01 or 0.001). No entropy coding for residues.

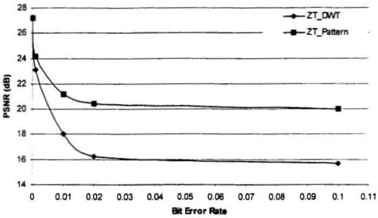


Figure 7.7 Average PSNR performance of ZT_Pattern as compared to ZT_DWT

on *Flower Garden*. Arithmetic coding is applied for residues.

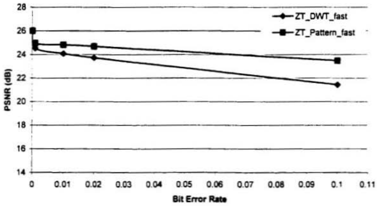
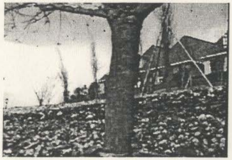
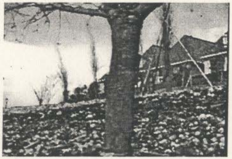


Figure 7.8 Average PSNR performance of ZT_Pattern as compared to ZT_DWT

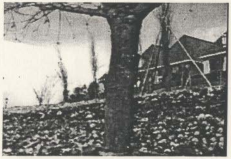
on *Flower Garden*. No entropy coding for residues.



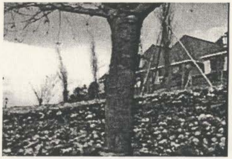
(a) Post-filtered result of Figure 7.3 (b)



(b) Post-filtered result of Figure 7.3 (d)



(c) Post-filtered result of Figure 7.4 (b)



(d) Post-filtered result of Figure 7.4 (d)

Figure 7.9 Post-processing results of applying a median filter on the reconstructed frames of Figure 7.3 (b) and (d), and Figure 7.4 (b) and (d) .

### 7.2.2 For Frames from *Football* Video

Frame 1 and 2 (shown in Figure 7.1) are chosen from *Football* sequence for the simulations.

Similar to section 7.2.1, Figure 7.10 to 7.13 show the reconstructed frames where the coded residue information suffers errors. Table 7.2 summarizes these reconstructed frames and corresponding residue coding techniques.

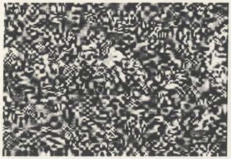
Figure 7.14 and 7.15 illustrate the average PSNR performance of video coding with different residue coding schemes on the virtual noisy channel. PSNR is the peak signal to noise ratio of the reconstructed frame. Simulations are repeated 70 times.

Figure 7.16 gives the post-processing results of applying a median filter on the reconstructed frames of Figure 7.10 (b), Figure 7.10 (d), Figure 7.11 (b), and Figure 7.11 (d).

Table 7.2 Summary of simulation results on reconstructed frames (*Football*)

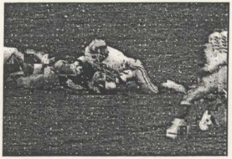
| Bit error rate | Entropy coding method | |
|---|---|---|
| | Arithmetic Coding | None |
| | ZT_DWT / ZT_Pattern | ZT_DWT_fast/ ZT_Pattern_fast |
| BER = 0.1 | Figure 7.10 | Figure 7.12 |
| BER = 0.02 | | |
| BER = 0.01 | Figure 7.11 | Figure 7.13 |
| BER = 0.001 | | |

The explanation of the abbreviations in the tables and figures are the same as in section 7.2.1.



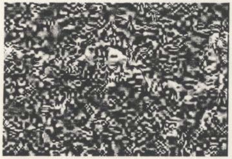
(a) ZT_DWT (BER = 0.1)

PSNR = 12.07 dB



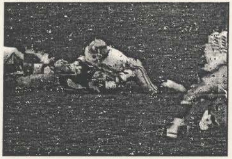
(b) ZT_Pattern (BER = 0.1)

PSNR = 19.92 dB



(c) ZT_DWT (BER = 0.02)

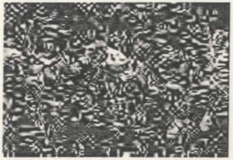
PSNR = 12.60 dB



(d) ZT_Pattern (BER = 0.02)

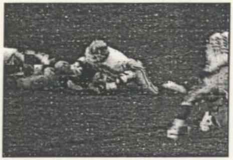
PSNR = 19.95 dB

Figure 7.10 Reconstructed frames from *Football* where the coded residue information suffers errors (BER = 0.1 or 0.02). Arithmetic coding is applied for residues.



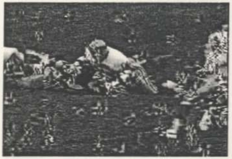
(a) ZT_DWT (BER = 0.01)

PSNR = 13.92 dB



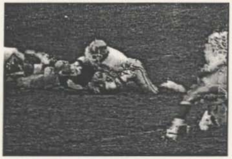
(b) ZT_Pattern (BER = 0.01)

PSNR = 20.31 dB



(c) ZT_DWT (BER = 0.001)

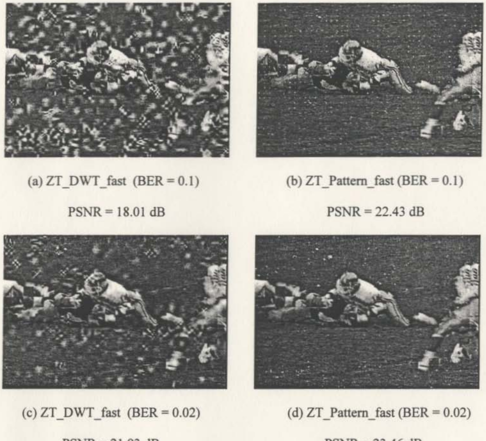
PSNR = 20.48 dB



(d) ZT_Pattern (BER = 0.001)

PSNR = 23.36 dB

Figure 7.11 Reconstructed frames from *Football* where the coded residue information suffers errors (BER = 0.01 or 0.001). Arithmetic coding is applied for residues.



(a) ZT_DWT_fast (BER = 0.1)

PSNR = 18.01 dB

(b) ZT_Pattern_fast (BER = 0.1)

PSNR = 22.43 dB

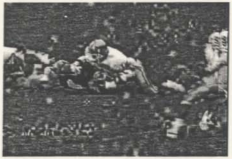
(c) ZT_DWT_fast (BER = 0.02)

PSNR = 21.93 dB

(d) ZT_Pattern_fast (BER = 0.02)

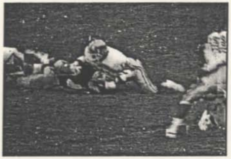
PSNR = 23.46 dB

Figure 7.12 Reconstructed frames from *Football* where the coded residue information suffers errors (BER = 0.1 or 0.02). No entropy coding for residues



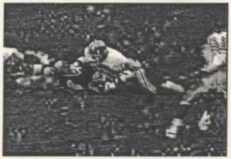
(a) ZT_DWT_fast (BER = 0.01)

PSNR = 22.42 dB



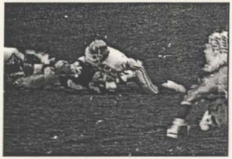
(b) ZT_Pattern_fast (BER = 0.01)

PSNR = 23.55 dB



(c) ZT_DWT_fast (BER = 0.001)

PSNR = 23.00 dB



(d) ZT_Pattern_fast (BER = 0.001)

PSNR = 23.70 dB

Figure 7.13 Reconstructed frames from *Football* where the coded residue information suffers errors (BER = 0.01 or 0.001). No entropy coding for residues.

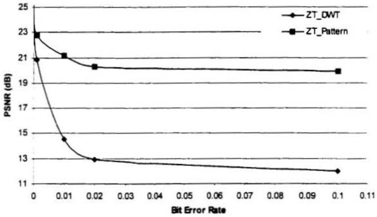


Figure 7.14 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Football*. Arithmetic coding is applied for residues.

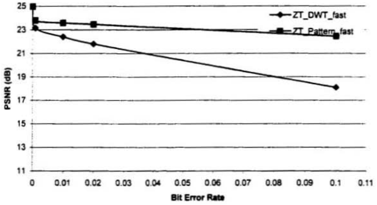
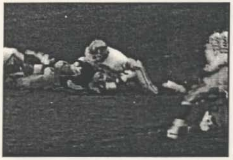


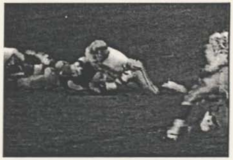
Figure 7.15 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Football*. No entropy coding for residues.

(a) Post-filtered result of Figure 7.10 (b)

(b) Post-filtered result of Figure 7.10 (d)



(c) Post-filtered result of Figure 7.11 (b)



(d) Post-filtered result of Figure 7.11 (d)

Figure 7.16 Post-processing results of applying a median filter on the reconstructed frames of Figure 7.10 (b) and (d), and Figure 7.11 (b) and (d).

## 7.3 Discussion of the Simulation Results in Section 7.2

In these simulations, the error resilience and concealment of the video coding scheme, which employs zerotree pattern coding for dealing with residues, is studied and compared with zerotree wavelet coding. Simulation results are reported in section 7.2. Discussion of these results is the task of this section.

### 7.3.1 Visual Effect of the Reconstructed Frames

Observing the reconstructed frames in Figure.7.3 to 7.6 and Figure 7.10 to 7.13, the appearance of the recovery results by different residue coding schemes (ZT_DWT/ZT_DWT_fast, ZT_Pattern/ZT_Pattern_fast) at different BER (0.1, 0.02, 0.01, 0.001) is very different.

The ZT_DWT results (such as Figure 7.3 (a)) contain lots of noise. The noise merges into an organic whole and causes ugly distortion. In the cases of high bit error rates (BER), the recovered picture is covered by such noise, and people cannot interpret the picture. Compared to the ZT_DWT results, the ZT_Pattern results (such as Figure 7.3 (b)) look much clearer. There is snow-like noise in the reconstructed picture. Even if it suffers heavy noise, people can still visualize the image.

If the entropy coding part is deleted from the residue coders, all the recovered frames look a little clearer. The visual difference between ZT_Pattern_fast and ZT_DWT_fast becomes less. However, the ZT_Pattern_fast results (such as Figure 7.5 (b)) are still the most acceptable of those achieved at the same BER and data rate.

### 7.3.2 Effect of Entropy Coding

Entropy coding is used to improve the coding efficiency. However, entropy coding causes error propagation. Specifically, in ZT_DWT, and ZT_Pattern, arithmetic coding is applied as part of the residue coding scheme. Arithmetic coding transforms input data into an output bit stream based on the placement of the current data value in the source probability distribution. If the coded residue data suffers errors, the decoder loses synchronization and its probability estimates are distorted. Recovery is unlikely and resynchronization is almost impossible. Without entropy coding, the decoder is able to recover.

### 7.3.3 Zerotree Approach

It has been confirmed in chapter 6 that the zerotree approach is the key factor for zerotree coding to achieve high coding efficiency. However, the built-in tree structure of the general zerotree approach and the implicit dependence of the significance map coding strategy make it sensitive to channel errors. Since the zerotree approach provides a compact multiresolution representation of significance maps and progressively transmits significance maps according to their importance, zerotrees allow the successful prediction of insignificant coefficients across scales to be represented as part of exponentially growing trees. Therefore, the quality of the recovered picture depends on correctly decoding the significance maps. The earlier an error occurs, the more it degrades the decoding quality. An error that occurs in the high level of the tree may spread out to the neighbouring levels and down to the next level.

The results in Figure 7.2 clearly illustrate the influence of errors on the zerotree approach. Because there is no wavelet transform or ordering pattern as pre-processing in the raw zerotree coding, lots of noise concentrates on the upper-left corner of a recovered picture.

Different from the raw zerotree coding case, in the recovery results for wavelet and pattern coding, the errors do not accumulate on one area. The errors are dispersed through the whole picture.

For zerotree wavelet coding, since each coefficient of the wavelet transform represents a spatial area of the original image, a single error in the encoded information corrupts the whole of that area after the inverse wavelet transform during the decoding procedure. This is in addition to the error propagation effect mentioned above.

For the zerotree pattern coding, since the original image is segmented into blocks for the use of ordering patterns and the tree is built based on such patterns, the errors do not accumulate on one spatially area after the inversion of the ordering patterns during the decoding pass. Furthermore, errors will propagate through spatially separated pixels rather than through spatial adjacent wavelet coefficients. Hence, in this case, the reconstructed frames have better visual appearance than the other two cases above.

Quantitative evaluation is done by comparing the peak signal to noise ratios (PSNR) of the reconstructed frames. Simulations are repeated 70 times. The comparison results are shown in Figure 7.7 and 7.8, and Figure 7.14 and 7.15.

Considering the entropy coding case, for *Flower Garden*, at a bit error rate of 0.02, ZT_Pattern outperforms ZT_DWT by about 4 dB (Figure 7.7). For *Football*, at a bit error rate of 0.02, ZT_Pattern outperforms ZT_DWT by about 7 dB (Figure 7.14). To summarise, comparison results show that the average PSNR performance of video coding with patterns outperforms coding with wavelets.

### 7.3.4 Error Concealment

The error concealment of the proposed scheme is studied in these simulations. The main purpose of error concealment is to mask the errors into the decoded image. As shown in Figure 7.9 and Figure 7.16, after applying a median filter on the recovery results of patterns, the noise in the reconstructed frames is removed. The cost is that the post-processing picture has a slightly blurred effect. The PSNR of the post-processing picture is sometimes higher than that before filtering, and sometimes lower. However, the visual effect of the post-processing picture is always improved.

The noise cannot be removed from the ZT_DWT/ZT_DWT_fast reconstructed frames and those using raw zerotree coding (Figure 7.2) by this simple median filter.

It can be seen that the application of simple post-processing provides additional error concealment for the pattern coding strategy.

### 7.3.5 Comments

In these simulations, the simulation of transmission on a noisy channel is accomplished by adding errors into the coded residue information with the coded motion vectors and the first 7 bytes (the header) of the coded residue stream protected. The reason is that the total bits occupied by the coded motion vectors, together with the header of the coded residues, are a relatively small amount as compared with the bits used by the coded residues (without the header). For instance, if *Flower Garden* (25 frames) is transmitted in an average bitrate 2.16 Mbits/s, then the coded residues (zerotree pattern coding) will cost 10553 bytes per frame whereas the coded motion vectiors (Huffman coding) and the header will use about 203 bytes per frame. In practical applications, the motion vectors and the header can be protected from channel noise by error control techniques (refer to chapter 2, section 2.3).

## 7.4 Simulations on Video Sequences

The simulations reported in section 7.2 are performed on two frames instead of an entire video sequence. In this section, some simulations are conducted on video sequences (*Flower Garden* and *Football*). Refering to section 6.2 and 7.1.1, the simulation procedures for video sequences are similar to before. In the encoder, the residues are coded by either the zerotree wavelet coder or the zerotree pattern coder ("Pattern-best") with entropy coding. Omitting the bits (0.0039 bpp) used for flags of patterns, the coding bit rate for each residue picture is 0.1 bpp (1056 bytes) for *Flower Garden* and 0.5 bpp (5280 bytes) for *Football*. Random errors are added to each coded residue picture (without header). However, in the receiver, the decoder has two strategies to recover the video frames: abandoning residues or including residues.

### 7.4.1 Decoding Strategy of Abandoning Residues

There are various mixed strategies where error detection or correction might be used as well as pattern coding. For instance, the residue information can be divided into blocks and additional codes are added into each block for detecting errors: if errors are detected, the decoder may throw away this block of residues; if no error is found, it can keep this part of residues. Such strategies result in additional redundancy information. However, in this thesis, there is no error detection or correction used for residue information, and only the motion vectors and the header of residues are protected from channel errors. Hence the above mixed strategies have not been studied. But due to the fact that the motion vectors and the header are protected by error control techniques, the receiver may detect errors if there are errors on the channel while transmitting the motion vectors and the header. Then the receiver can just throw away all the residue information and use only the motion vectors to obtain the recovered frames. In such a case, the recovered frames are the prediction pictures. The decoding procedures of this strategy are listed as below:

Step1: Decode the coded motion vectors. Abandon all the coded residue information (Suppose the receiver finds out that there are errors in the channel).

Step 2: Decode the first compressed frame as recovered frame 1.

Step 3: Set $n = 1$.

Step 4: Shift the blocks of the recovered frame n according to the corresponding motion vectors to obtain the prediction picture $(n+1)$. The prediction picture $(n+1)$ is the recovered frame $(n+1)$.

Step 5: If $n < N$ (Suppose the total frame number is $N$), $n = n + 1$, goto step 4.

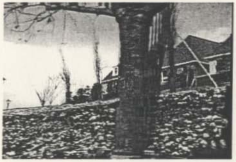
With this strategy, some simulations are conducted and some recovered frames from *Flower Garden* and *Football* are shown in Figure 7.17 and 7.18.

Recovered frame 2

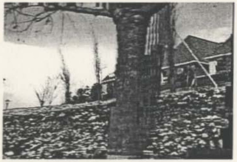
Recovered frame3

Recovered frame 4

Recovered frame5

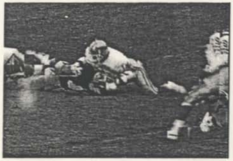


Recovered frame 6

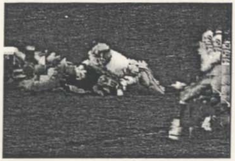


Recovered frame7

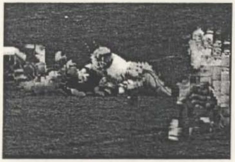
Figure 7.17 Recovered frames from *Flower Garden* with the decoding strategy of abandoning residues.



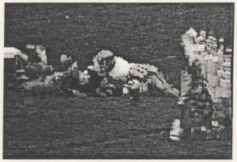
Recovered frame 2



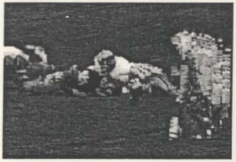
Recovered frame3



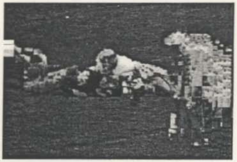
Recovered frame 4



Recovered frame5



Recovered frame 6



Recovered frame7

Figure 7.18 Recovered frames from *Football* with the decoding strategy of abandoning residues.

### 7.4.2 Decoding Strategy of Including Residues

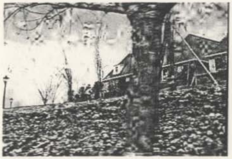
The decoding procedures with the strategy of including residues are the same as those in section 6.2. Two kinds of residue coding and decoding schemes are applied in this decoding strategy: zerotree wavelet coding and zerotree pattern (Pattern-best) coding. For each scheme, there are still two options: with entropy coding (arithmetic coding) or without entropy coding. A large number of exhaustive simulations have been done in this context, and some representative results are shown here. Table 7.3 and 7.4 list these results.

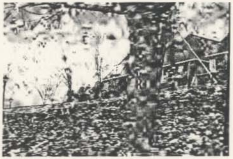
Table 7.3 Summary of simulation results on *Flower Garden*

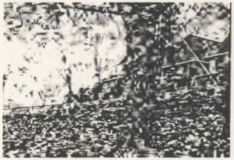
| Bit error Rate | Entropy coding method | |
|---|---|---|
| | Arithmetic Coding | None |
| | ZT_DWT / ZT_Pattern | ZT_DWT_fast / ZT_Pattern_fast |
| BER = 0.1 | | Figure 7.19 |
| BER = 0.01 | | Figure 7.20 |
| BER = 0.001 | Figure 7.21 | |

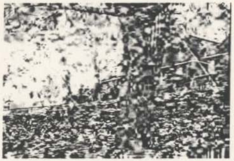
Table 7.4 Summary of simulation results on *Football*

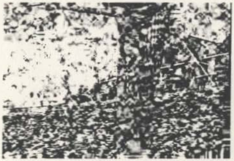
| Bit error Rate | Entropy coding method | |
|---|---|---|
| | Arithmetic Coding | None |
| | ZT_DWT / ZT_Pattern | ZT_DWT_fast / ZT_Pattern_fast |
| BER = 0.1 | | Figure 7.22 |
| BER = 0.002 | | Figure 7.23 |
| BER = 0.0002 | Figure 7.24 | |

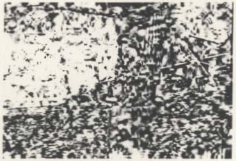

Reconstructed frame 2


Reconstructed frame3


Reconstructed frame 4

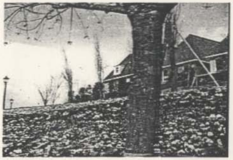

Reconstructed frame5


Reconstructed frame 6

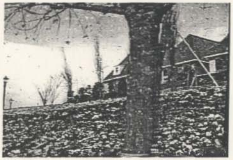

Reconstructed frame7

(a) ZT_DWT_fast

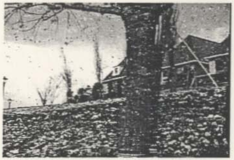
Figure 7.19 Recovered frames from *Flower Garden* where the coded residue information suffers errors (BER = 0.1). No entropy coding is applied for residues.



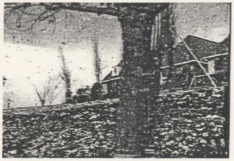
Reconstructed frame 2



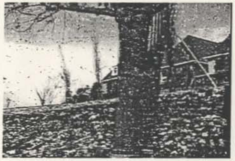
Reconstructed frame3



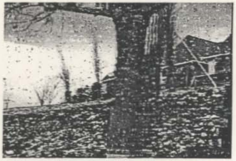
Reconstructed frame 4



Reconstructed frame5



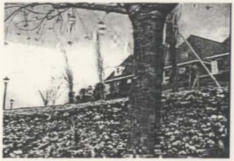
Reconstructed frame 6

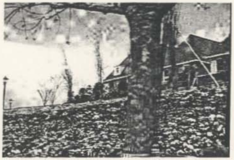


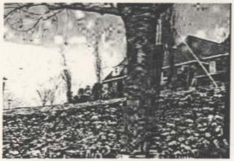
Reconstructed frame7

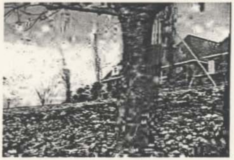
(a) ZT_Pattern_fast

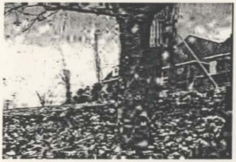
Figure 7.19 Recovered frames from *Flower Garden* where the coded residue information suffers errors (BER = 0.1). No entropy coding is applied for residues. (Continued)

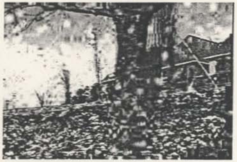

Reconstructed frame 2


Reconstructed frame3


Reconstructed frame 4

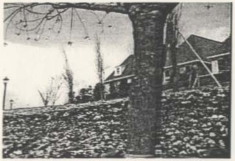

Reconstructed frame 5


Reconstructed frame 6

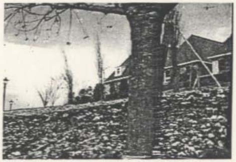

Reconstructed frame 7

(a) ZT_DWT_fast

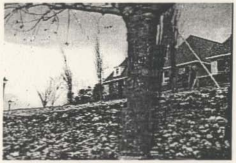
Figure 7.20 Recovered frames from *Flower Garden* where the coded residue information suffers errors (BER = 0.01). No entropy coding is applied for residues.



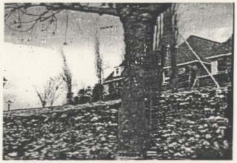
Reconstructed frame 2



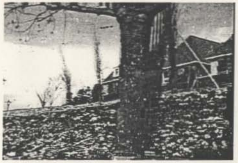
Reconstructed frame3



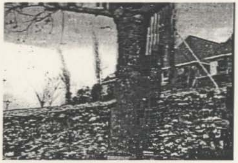
Reconstructed frame 4



Reconstructed frame5



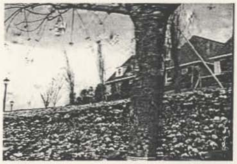
Reconstructed frame 6



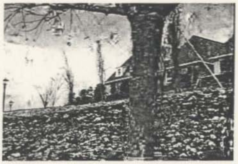
Reconstructed frame7

(b) ZT_Pattern_fast

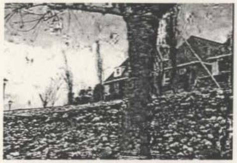
Figure 7.20 Recovered frames from *Flower Garden* where the coded residue information suffers errors (BER = 0.01). No entropy coding is applied for residues. (Continued)



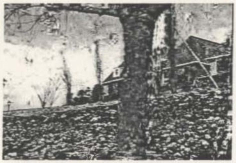
Reconstructed frame 2



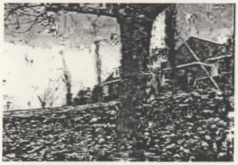
Reconstructed frame3



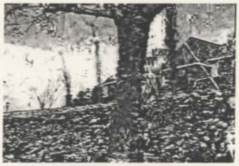
Reconstructed frame 4



Reconstructed frame 5



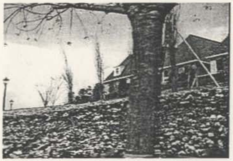
Reconstructed frame 6



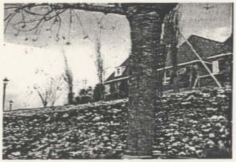
Reconstructed frame 7

(a) ZT_DWT

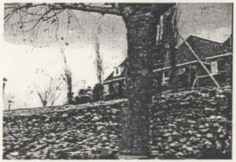
Figure 7.21 Recovered frames from *Flower Garden* where the coded residue information suffers errors (BER = 0.001). Arithmetic coding is applied for residues.



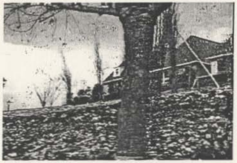
Reconstructed frame 2



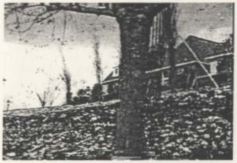
Reconstructed frame3



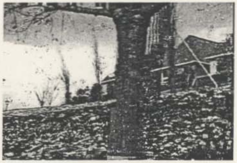
Reconstructed frame 4



Reconstructed frame5



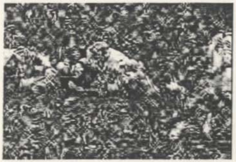
Reconstructed frame 6



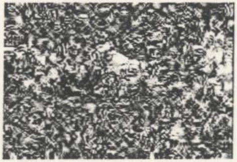
Reconstructed frame7

(b) ZT_Pattern

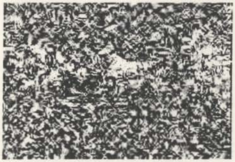
Figure 7.21 Recovered frames from *Flower Garden* where the coded residue information suffers errors (BER = 0.001). Arithmetic coding is applied for residues. (Continued)



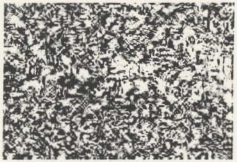
Reconstructed frame 2



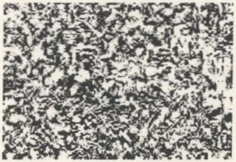
Reconstructed frame3



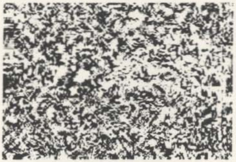
Reconstructed frame 4



Reconstructed frame5



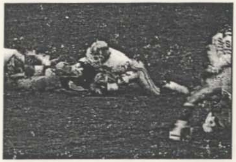
Reconstructed frame 6



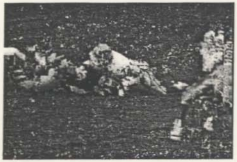
Reconstructed frame7

(a) ZT_DWT_fast

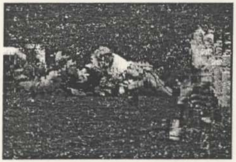
Figure 7.22 Recovered frames from *Football* where the coded residue information suffers errors (BER = 0.1). No entropy coding is applied for residues.



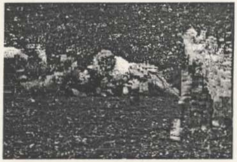
Reconstructed frame 2



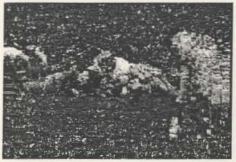
Reconstructed frame3



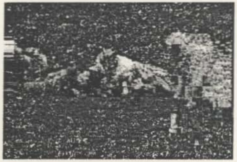
Reconstructed frame 4



Reconstructed frame5



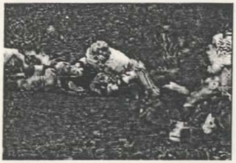
Reconstructed frame 6

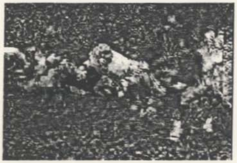


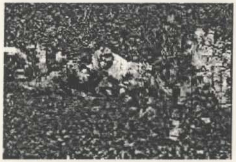
Reconstructed frame7

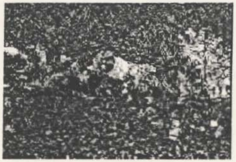
(b) ZT_Pattern_fast

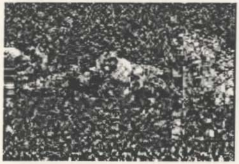
Figure 7.22 Recovered frames from *Football* where the coded residue information suffers errors (BER = 0.1). No entropy coding is applied for residues. (Continued)

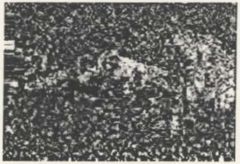

Reconstructed frame 2


Reconstructed frame3


Reconstructed frame 4

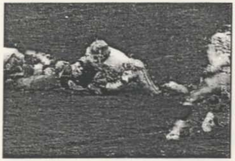

Reconstructed frame 5


Reconstructed frame 6

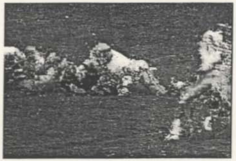

Reconstructed frame 7

(a) ZT_DWT_fast

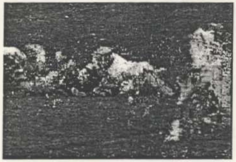
Figure 7.23 Recovered frames from *Football* where the coded residue information suffers errors (BER = 0.002). No entropy coding is applied for residues.



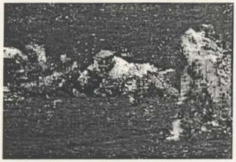
Reconstructed frame 2



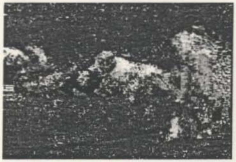
Reconstructed frame3



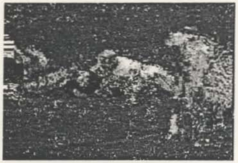
Reconstructed frame 4



Reconstructed frame5



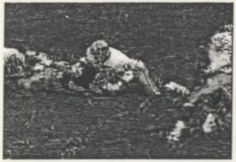
Reconstructed frame 6



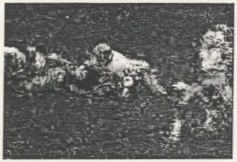
Reconstructed frame7

(b) ZT_Pattern_fast

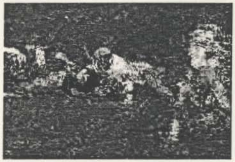
Figure 7.23 Recovered frames from *Football* where the coded residue information suffers errors (BER = 0.002). No entropy coding is applied for residues. (Continued)



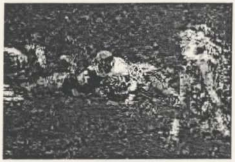
Reconstructed frame 2



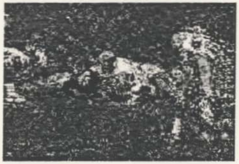
Reconstructed frame3



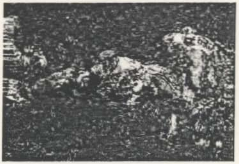
Reconstructed frame 4



Reconstructed frame 5



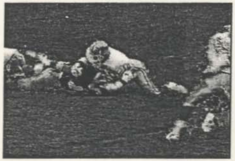
Reconstructed frame 6



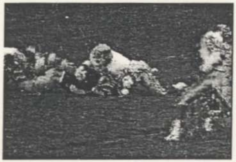
Reconstructed frame 7

(a) ZT_DWT

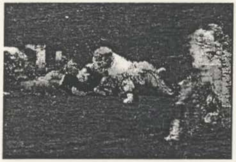
Figure 7.24 Recovered frames from *Football* where the coded residue information suffers errors (BER = 0.0002). Arithmetic coding is applied for residues.



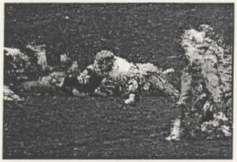
Reconstructed frame 2



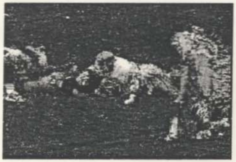
Reconstructed frame3



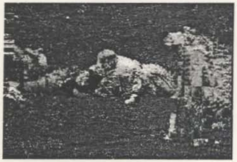
Reconstructed frame 4



Reconstructed frame5



Reconstructed frame 6



Reconstructed frame7

(b) ZT_Pattern

Figure 7.24 Recovered frames from *Football* where the coded residue Information suffers errors (BER = 0.0002). Arithmetic coding is applied for residues. (Continued)

### 7.4.3 Discussion

From the results shown in Figure 7.17 to 7.24, it can be seen that the quality of the recovered frames decreases with position in the sequences. This quality drop disturbs the visual impression in different ways depending on which kind of decoding strategy was applied.

For the decoding strategy of abandoning residues, the recovered images are only prediction pictures, there are no residues added to refine them, and also there is no channel noise to affect them. In these recovered images, the parts with less motion are reconstructed well, such as the ground in *Football* and some flowers in *Flower Garden.* However, the motion parts in these recovered frames, such as the tree and the house in *Flower Garden* and the players in *Football*, have strong block effect. That is because of the block-based nature of the motion estimation and motion compensation. For *Flower Garden* (Figure 7.17), which contains large slow moving detail information, the block effect is not remarkable in the beginning (Figure 7.17), but it appears more and more frequently and more and more black blocks appear with the increase of the recovered frames. For *Football* (Figure 7.18), which includes large motion, the block effect is clearly visible in the first recovered frame, and then becomes more and more serious so that it is difficult to finger out if there is a person in the right part of the recovered image.

For the decoding strategy of including residues, it can be seen that the decoding strategy with patterns always performs better than that with wavelets (Figure 7.19 – 7.24). For very noisy channels (BER = 0.1), the reconstructed frames with patterns are relatively clear in the beginning, and then gradually degrade, whereas the wavelet method produces unrecognizable decoded frames rapidly. With the decrease of BER, the block effect in the recovered frames of pattern decoding scheme becomes weak, and the recovered frames become more and more clear.

Comparing the two strategies, in the very noisy channels, the performances of the decoding strategy of abandoning residues and the strategy of pattern method are comparable. However, in the less noisy channels, the pattern residue scheme performs better than that throwing away all the residues.

Note that the motion vectors are obtained in the encoding process, and the residues are involved when the encoder generates the motion vectors (Refer to section 6.2). If the decoder (in the receiver) throws away all residue information and reconstructs the frames (prediction pictures) using only motion vectors, then the quality of the prediction pictures will be degraded. The more residues that are discarded, the worse will be the quality of the prediction pictures.

## 7.5 Summary

A large number of simulation results were reported in this chapter for evaluating the channel error resilience and concealment of the video coding scheme proposed in the thesis. A detailed analysis about these results was presented as well. Clearly, the pattern coder is far superior to the wavelet coder in this context.

# Chapter 8

# Conclusions and Future Work

## 8.1 Conclusions

This thesis has discussed the strategy of coding motion picture residues without applying transforms and explored the efficiency of the proposed video coding scheme and the possibilities for error resilience and concealment.

From many simulations and simulation results, the following conclusions have been reached:

- The proposed video coding scheme is more efficient than MPEG.
- Zerotree pattern coding performs close to the zerotree wavelet state-of-the-art and much better than MPEG in error free channels.
- Zerotree pattern coding is superior to zerotree wavelet coding in the case of noisy channel conditions.

It has been demonstrated that the video coding strategy proposed in this thesis achieves good trade-offs between compression performance and channel error resilience and robustness.

For computational complexity, whether with or without entropy coding, for the encoder, zerotree pattern coding achieves about 5 percent improvement in computational efficiency than zerotree wavelet method, and for the decoder, the improvement is about 15 percent.

The major contribution of the thesis is to put forward the particularity of the prediction image residues, which indicates that they may be compressed differently from still images. This thesis has also contributed towards the development of applying ordering patterns rather than wavelets in zerotree residue coding, which results in error propagation through spatially separated pixels rather than spatially adjacent wavelet coefficients when residues suffer errors. The implemented residue coding scheme together with the half-pixel accuracy motion estimation and motion compensation technique and Huffman coding for motion vectors form a novel video coding strategy.

As for MPEG dealing with video coding in error-free channels, the expected applications for the novel video coding strategy implemented in this thesis will be in areas of the error-resilient transmission of video data over communication channels, especially mobile wireless channels, where the user may require the video coder to provide high compression performance over error free conditions and prefer to obtain economically at least recognizable recovered pictures on noisy channels.

## 8.2 Future Work

There is still some work that needs to be done in the future.

Firstly, more work is required to explore the effects of different types of errors, including burst errors (such as limited length bursts and longer bursts), on video transmission by building noisy channel models. Generally, there are three kinds of channel models: discrete memoryless channel (DMC) model, binary symmetric channel (BSC) model, and Gaussian channel model.

Secondly, it is necessary to study the degree of protection required for motion vector information. In this thesis, all the motion vectors are protected in case of channel errors. To reduce the redundancy information, it is worth to find out whether it is possible to protect partial motion vector information.

Thirdly, the alternatives for error concealment post-processing, such as the use of non-linear filters, need to be investigated. In this thesis, only the visual effect of applying simple median filter as post-processing was studied.

Fourthly, coding the colour residues using the same methods as the luminance residues need be studied.

## References

Cheng, N. T. and Kingsbury, N. G. (1992). "The ERPC: An Efficient Error-Resilient Technique for Encoding Positional Information or Sparse Data." *IEEE Transactions on Communications*, Vol. 40, no. 1, pp. 140-148.

Ferguson, T. J. and Rabinowitz, J. H. (1984). "Self-Synchronizing Huffman Codes." *IEEE Transactions on Information Theory*, Vol. IT-30, no. 4, pp. 687-693.

Furht, B., Greenberg, J., and Westwater R. (1997). *Motion Estimation Algorithms for Video Compression*, Kluwer Academic Publishers, 101 Philip Drive, Assinippi Park, Norwell, Massachusetts 02061, 163 p.

Gall, D.L. (1991). "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of the ACM*, Vol. 34, no. 4, pp. 46-58.

MacDonald, N. (1992). "Transmission of Compressed Video over Radio Links," *Proceedings of SPIE International Conference on Visual Communications and Image Processing*, Boston, MA, U.S.A., pp. 1484-1488.

Man, H. and Kossedtini, F. (1997). "Robust EZW Image Coding for Noisy Channels," *IEEE Signal Processing Letters*, Vol. 4, no. 8, pp. 227-229.

Montgomery, B. L. and Abrabams, J. (1986). "Synchronizing of Binary Source Codes," *IEEE Transactions on Information Theory*, Vol. IT-32, no. 6, pp. 849-854.

Redmill, D. W. and Kingsbury, N. G. (1993). "Improving the Error Resilience of Entropy Coded Video Signals," *Proceedings of International Conference on Image Processing: Theory and Applications*, San Remo, Italy, pp. 67-70.

Redmill, D. W. and Kingsbury, N. G. (1996). "The EREC: An Error-Resilient Technique for Coding Variable-Length Blocks of Data," *IEEE Transactions on Image Processing*, Vol. 5, no. 4, pp. 565-574.

Said, A and Pearlman, W. A. (1993). "Image Compression Using the Spatial Orientation Tree," *Proceedings of IEEE International Symposium on Circuits and Systems*, Chicago, IL, U.S.A., Vol. 4, pp. 279-282.

Said, A. and Pearlman, W. A. (1996). "A New Fast and Efficient Image Codec Based on Set Partitioning Hierarchical Trees," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, no. 3, pp. 243-250.

Shapiro, J. M. (1993). "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Transactions on Signal Processing*, Vol. 41, no. 12, pp. 3445-3462.

Sherwood, P. G. and Zeger, K. (1997). "Progressive Image Coding on Noisy Channels," *Proceedings of IEEE Data Compression Conference*, Snowbird, Utah, U.S.A., pp 72-81.

Umbaugh, S. E. (1998). *Computer Vision and Image Processing: A Practical Approach Using CVIPTools*, Prentice-Hall PTR, Upper Saddle River, New Jersey 07458, 504p.

# Appendices

## Appendix I: Overall Transmission Data Rate of 0.53 bpp

The performances of zerotree pattern coding and zerotree wavelet coding over noisy channel at an overall data rate of 0.53 bpp are attached here.

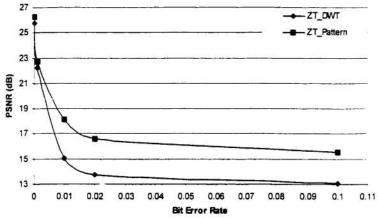


Figure I.1 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Flower Garden*. Arithmetic coding is applied for residues.

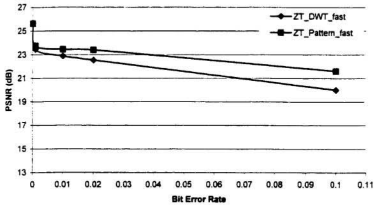


Figure I.2 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Flower Garden*. No entropy coding for residues.

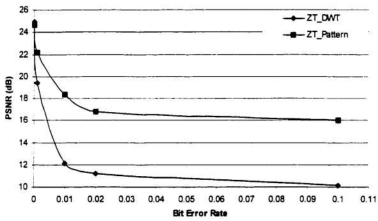


Figure I.3 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Football*. Arithmetic coding is applied for residues.

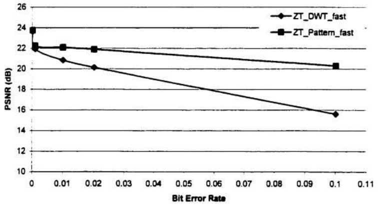


Figure I.4 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Football*. No entropy coding for residues.

## Appendix II: Overall Transmission Data Rate of 0.23 bpp

The performances of zerotree pattern coding and zerotree wavelet coding over noisy channel at an overall data rate of 0.23 bpp are attached here.

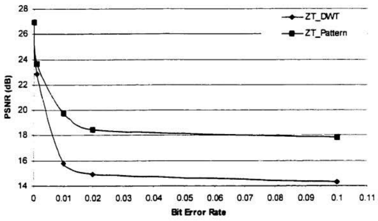


Figure II.1 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Flower Garden*. Arithmetic coding is applied for residues.

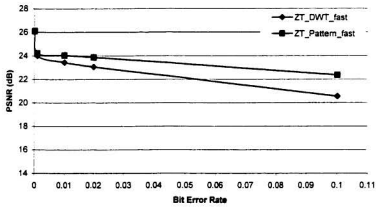


Figure II.2 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Flower Garden*. No entropy coding for residues.

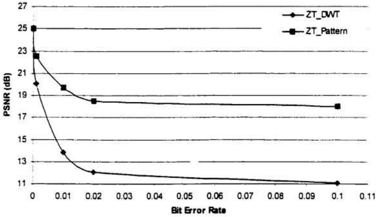


Figure II.3 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Football*. Arithmetic coding is applied for residues.

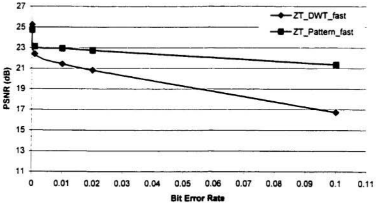


Figure II.4 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Football*. No entropy coding for residues.

## Appendix III: Overall Transmission Data Rate of 0.08 bpp

The performances of zerotree pattern coding and zerotree wavelet coding over noisy channel at an overall data rate of 0.08 bpp are attached here.

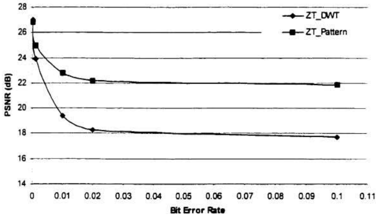


Figure III.1 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Flower Garden*. Arithmetic coding is applied for residues.

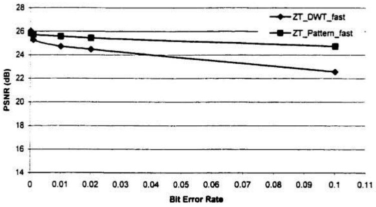


Figure III.2 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Flower Garden*. No entropy coding for residues.

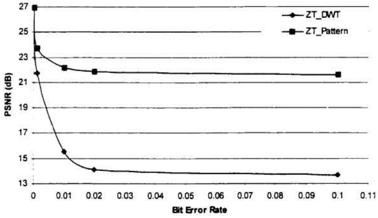


Figure III.3 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Football*. Arithmetic coding is applied for residues.

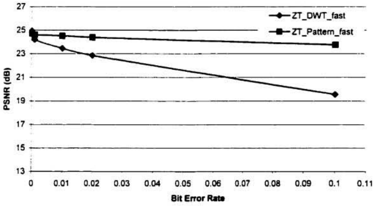


Figure III.4 Average PSNR performance of ZT_Pattern as compared to ZT_DWT on *Football*. No entropy coding for residues.

# Bibliography

Bylanski, P. and Ingram D. G. W. (1980). *Digital Transmission System*, Peter Peregrinus Ltd., Stevenage, Herts SG1 1HQ, 431p.

Clark, R. J. (1996). *Digital Compression of Still Images and Video*, Academic Press Inc., San Diego, California 92101, 449p.

Jain, A. K. (1989). *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, New Jersey 07632, 569p.

Lindner, S. (1995). "Video Coding Using Motion Palettes," *M.Eng. thesis*, University of Waterloo, Waterloo, Ontario, Canada.

Teuber, J. (1993). *Digital Image Processing*, Prentice-Hall International (UK) Ltd., Hemel Hempstead, Hertfordshire HP2 7EZ, 263p.

Weiss, L. G. (1994). "Wavelets and Wideband Correlation Processing," *IEEE Signal Processing Magazine*, Vol. 11, no. 1, pp. 13-32.