Listless Zerotree Image and Video Coding

Wen-Kuo Lin
B.E.(Hons)
The University of Adelaide, Australia

Thesis submitted for the degree of
Doctor of Philosophy
in the
Department of Electrical and Electronic Engineering
The University of Adelaide

2001

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Zerotree coding algorithms are known to be very efficient in coding wavelet transform images, in terms of coding complexity. However, zerotree coding algorithms suffer from a large coding memory requirement. Therefore, in order to solve this problem, this thesis proposes a zerotree coding algorithm called listless zerotree coding for image and video compression. The proposed coding algorithm significantly reduces the coding memory requirement and has simpler coding complexity than any other published zerotree coding algorithm. Therefore, the proposed algorithm is more suitable for hardware implementation than others.

We begin our discussion by looking at rate-distortion optimization, which is the fundamental consideration behind image and video compression. That is, the goal of image and video compression is to use a small amount of bit-budget to encode as much information as possible for a given distortion. To this end, we need to identify what sort of visual information is more important to our visual system, and what sort of information is not. Therefore, in this thesis we also investigate the characteristics of the human visual system that are related to image and video compression, and how we can apply those characteristics to the compression scheme.

In a large portion of this thesis, we focus our discussion on the proposed listless zerotree coding algorithm. We first discuss the application of the listless zerotree coding algorithm in still color image compression. The discussion includes the mapping of visual system characteristics to the coding algorithm, details of algorithm implementation, and the implementation options. After that, we discuss the application of listless zerotree coding algorithm in color video compression. The discussion includes 3D wavelet transform and 3D zerotree data structure, as well as implementation details and options. Based on the experimental results, we can see that the proposed listless zerotree coding algorithm performs just as well as the benchmark zerotree algorithms and compression standards.

At the end of this thesis, we suggest two possible extensions for improving the coding performance and adding some functionalities to our listless zerotree algorithm.

Statement of Originality

I hereby declare that this work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text.

I give consent to this copy of my thesis, when deposited in the University Library, being available for loan and photocopying.

Wen-Kuo Lin
30 November 2001

Acknowledgments

I have received help and support from many people throughout the course of my Ph.D. studies. Without their help, my work would have been more difficult to complete.

Firstly, I would like to express my gratitude to my supervisors, Professor Neil Burgess, Mr. Michael J. Liebelt, and Dr. Alireza Moini. I would like to thank Professor Neil Burgess for supervising most of my Ph.D. work and for providing me with financial support during the early stages of my candidature and for all of the academic conferences during the period of my studies. He also broadened my view in the way of doing research, as well as the beer drinking culture. I would also like to thank Dr. Alireza Moini for his supervision of my work after Professor Neil Burgess left the Department. I would specially like to thank Mr. Michael J. Liebelt for supervising my thesis writing during the last critical stages of my Ph.D. studies. Without his supervision my studies would have been far from complete.

Secondly, I would like to thank my friends in the old Digital Lab. Their encouragement was a source of inspiration for me. I would also like to thank my fellow postgraduates in the Department for the invaluable discussions, assistance, as well as countless after-hour entertainments and weekend sushi gatherings, and for making my studies in this Department an enjoyable experience. I should also thank all my friends in Adelaide for making my stay here a cheerful and an unforgettable experience. I am thankful to my friends who did their best to make my thesis more readable.

Finally, I would like to thank my parents in Taiwan for supporting me to pursue my Ph.D. studies in Australia. Without their support in the first place, I would have never had a chance to begin postgraduate studies.

Publications

1. **Wen-Kuo Lin** and Neil Burgess, " Listless Zerotree Coding For Color Images", 32nd Asilomar Conference on Signal, System, and Computer, Monterey CA, November 1998

2. **Wen-Kuo Lin** and Neil Burgess, "Low Memory Color Image Zerotree Coding", IEEE IDC'99, Adelaide SA, February 1999

3. **Wen-Kuo Lin**, Neil Burgess, Brian Wai-Him Ng, and Abedesselam Bouzerdoum, "Reduced Memory Zerotree Coding Algorithm for Hardware Implementation", IEEE ICMCS'99, Florence, Italy, June 1999

4. **Wen-Kuo Lin** and Neil Burgess, "A Low Memory Video Compression Algorithm for Hardware Implementation, IEEE MMSP'99, Copenhagen, Denmark, September 1999

5. **Wen-Kuo Lin** and Neil Burgess, "3D Listless Zerotree Coding for Low Bit Rate Video", IEEE ICIP'99, Kobe, Japan, October 1999

6. **Wen-Kuo Lin**, Alireza Moini and Neil Burgess, "Wavelet Transform Based Video Content Extraction for 3D Wavelet Video Coding", SPIE VCIP'00, Perth, Australia, June 2000

7. **Wen-Kuo Lin**, Alireza Moini and Neil Burgess, "Listless Zerotree Coding Using Raster Tree Search", IEEE TENCON'01, Singapore, August 2001

8. **Wen-Kuo Lin**, Alireza Moini and Neil Burgess, "Tree-Pruning Listless Zerotree Coding", IEEE ICME'01, Tokyo, Japan, August 2001

Symbols

B	branch set
$C(i,j)$	wavelet coefficient at coordinate (i,j)
$C(k,i,j)$	wavelet coefficient at coordinate (k,i,j)
$D(i,j)$	descendant coefficient set of $C(i,j)$
$D(k,i,j)$	descendant coefficient set of $C(k,i,j)$
F_C	significant map for wavelet coefficients
F_D	significant map for descendant coefficient set
F_T	significant map for tree branches
G	analysis high-pass filter
$\tilde{G}$	synthesis low-pass filter
H	analysis low-pass filter
$\tilde{H}$	synthesis low-pass filter
HH	wavelet subband containing diagonally oriented information
HL	wavelet subband containing horizontally oriented information
L	leaf set
LH	wavelet subband containing vertically oriented information
LL	wavelet subband containing lower frequency information
M or n	number of rows
N or n	number of columns
$O(i,j)$	child coefficient set of $C(i,j)$
$O(k,i,j)$	child coefficient set of $C(k,i,j)$
R	root coefficient set
T	number of frames
α	visual angle
i	row index
j	column index
k	frame index
m_r	row size of LL subband

n_r	column size of LL subband
f_r	frame numbers of LL subband
n	bit-layer index

Abbreviations

3DLZC	three dimensional listless zerotree coding
CIF	common intermediate format
CFF	critical fusion frequency
CIE	Commission Internationale de I'Eclairage
CSF	contrast sensitivity function
DCT	discrete cosine transform
EZW	embedded zerotree wavelet
FIR	finite impulse response
GOF	group of frame
HPF	high-pass filter
HVS	human visual system
IFS	iterated function system
ITU	International Telecommunication Union
JND	just-noticeable distortion
KLT	Karhunen-Loeve transform
LGN	lateral geniculate nucleus
LPF	low-pass filter
LSB	least significant bit
LZC	listless zerotree coding
ME/C	motion estimation and compensation
MSB	most significant bit
MSE	mean square error
MTF	modulation transfer function
PSNR	peak signal to noise ratio
QCIF	quarter common intermediate format
SPD	spectral density function
SPIHT	set partitioning in hierarchical trees
TPLZC	tree-pruning listless zerotree coding

VQ	vector quantization
bpp	bit per pixel
cpd	clocks per degree
fps	frame per second
kps	kilobits per second

Chapter 1

Introduction

1.1 Some Thoughts

Beautiful pictures are worth thousands of words. In recent years the Internet and telecommunication boom has brought us from the traditional text and voice communication to a new dimension of multimedia communication, mostly involving images and videos. However, those images and videos not only are worth thousands of words, but also cost millions of bytes of data traffic or computer storage. For example, a raw Common Intermediate Format (CIF) video frame requires 297KB of storage space, so one second of CIF video with the frame rate of 30 frames per second (fps) will require about 8.7 MB of storage space. Consequently, the dramatically large data size or high data traffic is the price we have to pay when shifting from conventional communication to multimedia communication.

Data communication between two points is achieved either by wired physical cables, or wireless electromagnetic waves. However, the data transmission rate is limited by the bandwidth of the wire or wireless distribution channels. Therefore, trying to transmit the raw digital image or video data through a limited-bandwidth distribution channel will be very inefficient. For example, a distribution channel having a bandwidth of 256 kilobits per second (kps) can only transmit 1.3 fps of uncompressed monochrome Quarter CIF (QCIF) video.

Besides transmitting through distribution channels, data communication can also be carried out by saving the data to a storage device at one location and retrieving the data from the storage device at another. Similar to transmission channels, storage devices also have a limited capacity, so it is not economical to store uncompressed image and video data in a limited-capacity storage device. For instance, a 650 MB compact disk (CD) can only store about 2240 uncompressed CIF video frames, or an equivalent of a 74-second video at 30 fps.

Therefore, in order to improve the transmission and storage efficiency, raw image and video data will have to be compressed to reduce their size. After compression, more image and video data can then be transmitted or stored, which in turn increases the amount of information communicated.

1.2 Foundations

A simple image and video data transmission model is shown in Figure 1.1. The transmitter takes and encodes the input image or video data at the source, and transmits the encoded bit-stream through the channel. The receiver takes the bit-stream and performs the inverse operation to the transmitter to produce the output image and video data at the destination. It is convenient to separate the function of the transmitter into *source coding* and *channel coding*. Source coding is aimed at determining what information is important, and how to effectively represent this information, while channel coding is aimed at determining how to transmit the source information, so that the errors caused by the channel noise can be minimized.

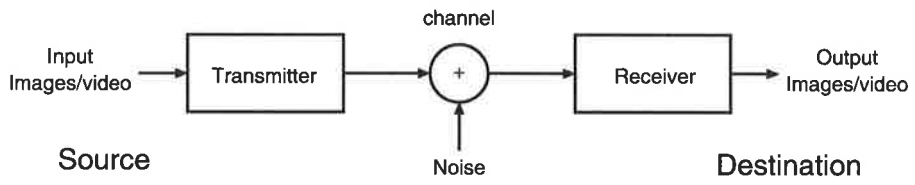


Figure 1.1: A simple image/video data transmission model

An ideal case for this transmission model is that both input and output data are identical. In the real application, channel noise is always present, regardless of the channel types, so there is some possibility that the image or video data may be corrupted during transmission. However, Shannon, in his *Fundamental Theorem for a Discrete Channel with Noise* [1], stated that

> "It is possible to send information at the rate C through the channel with as small a frequency of errors or equivocation as desired by proper encoding."

Shannon also defined the signal *Entropy*, which is the minimum number of bits required to encode a given signal. The signal Entropy is given by

$$H = -\sum_{i=1}^{N} P(i) log_2(P(i)), \tag{1.1}$$

where N is the number of independent source symbols and $P(i)$ is the probability of occurrence of symbol i. Assuming that the $Entropy$ of source symbols is H symbols per second and the channel capacity is C symbols per second, if a suitable *channel code* is used to encode arbitrary source symbols and satisfies $H \leq C$, then those symbols can be transmitted via an arbitrary channel with the best possible quality. However, we will focus the scope of this thesis on source coding, and we will assume the transmission channels are lossless in this thesis.

The image source coding can be lossless or lossy: in the former case the operation is invertible, and the image is perfectly reconstructed; while in the later case the operation is not invertible, and an approximate version of the image is produced. Since the size of video data is enormous compared to that of images, it is only practical to compress video with loss. In this thesis we will only consider lossy image and video compression, due to its wider practical applications.

The theoretical discipline behind lossy compression is the *rate-distortion theory* [2, 3], which optimizes the compressed bit-rate (R) and the distortion (D) from the viewpoint of *information theory*. A typical rate-distortion function of a source coding scheme is shown in Figure 1.2. Assuming constant source data, the maximum acceptable distortion is D_{max}, and the rate with-

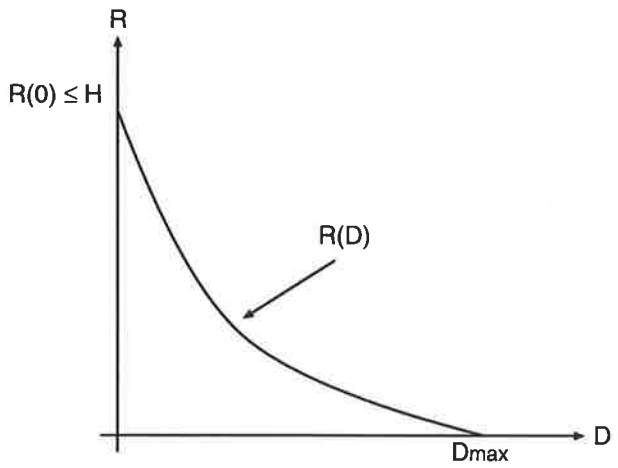


Figure 1.2: A typical rate distortion function

out distortion is $R(0)$, which is equal to or smaller than the source entropy H. The function $R(D)$ determines the bit-rate at which the source generates information. It suggests that in order to reduce the source bit-rate, the fidelity of source image or video will have to be sacrificed. That is, source image and video will have to be encoded at a bit-rate that is lower than their entropy. However, the bit-rate should be subject to the constraint that the viewer can tolerate the average distortion D, so the compression scheme should also attempt to minimize the distortion of the reconstructed information. Therefore, the major goal of image and video compression is

to minimize the source bit-rate, but at the same time also minimize the distortion for a given rate. The trade-off between rate and distortion is often referred as *rate-distortion optimization*, which can be formulated as

$$\min_{D} R(D) \quad \text{or} \quad \min_{R} D(R). \tag{1.2}$$

Intuitively, the rate-distortion can be optimized by using less of the coding bit-budget to code the information that has more contribution in minimizing the distortion on the reconstructed visual data. In order to achieve a better rate-distortion optimization, we then have to ask ourselves two essential questions:

Question 1.

What types of information are most important for minimizing distortion?

Question 2.

How can we effectively encode those important types of information using less of the coding bit-budget?

As the reconstructed image and video are to be viewed by the human viewers, we should refer to the *distortion* in Question 1 as the visual imperfection of the perceived image or video quality. Therefore, the more important types of information for minimizing the distortion will be those that are perceptually more important to our visual system. After identifying this visually important information, we can then apply a *bit-allocation* scheme to encode the information. The performance of an image and video coding algorithm will then mainly depend on how good the bit-allocation scheme is. However, finding an effective bit-allocation scheme is still one of the most active research topics in the image compression societycommunity, and an optimal scheme is yet to be found.

1.3 The Scope of the Thesis

Although the development of the Internet and mobile communications has opened up a new field for multimedia data communications, the limited transmission bandwidth is still a critical issue for such applications. Therefore, from the viewpoint of transmission efficiency, multimedia data will have to be transmitted at *low bit-rate*, a transmission rate which is much lower than the signal entropy.

Among all the compression schemes, only the *wavelet compression scheme* can achieve a satisfactory coding quality at low bit-rate, whereas the current image and video compression standards will simply fail to maintain an acceptable coding quality, if the bit-rate is too low.

Among the wavelet compression schemes, the *zerotree algorithm* not only has a comparatively low coding complexity, but also has the desirable quality scaling feature, which is suitable for heterogeneous channel capacity. The quality scaling feature is also known as *embededness*. In spite of those advantageous features, all zerotree image and video coding algorithms suffer from a severe high coding memory requirement for maintaining the coding lists. Besides, the coding complexity of zerotree coding algorithms can be further simplified for both software and hardware implementation.

Therefore, the major goal of this thesis is to derive a zerotree image and video coding algorithm that has a much lower coding memory requirement and a much simpler coding complexity. In this thesis, we will also focus the discussion on image and video compression related topics: the characteristics of the human visual system for identifying visually important information; visual data correlation and redundancy for obtaining a more compact visual data representation; an image and video coding scheme which exploits human visual characteristics and data redundancy.

1.4 Contributions of the Thesis

The contributions made in this thesis are as follows:

Firstly, this thesis provides a review of the psychophysical and physiological aspects of the human visual system. In particular, this thesis critically reviews the human vision characteristics that can be exploited for image and video compression.

Secondly, this thesis gives a critical discussion of visual data correlations, and suggests coding processes to reduce the redundancies caused by data correlation.

Thirdly, this thesis provides an overview of general visual data compression schemes, and the methods used to assess compression quality.

Fourthly, this thesis critically reviews the current zerotree image and video compression algorithms, and identifies the drawbacks of those algorithms to open a gap for the proposed algorithm of this thesis.

Finally, the major contribution of this thesis is to propose a zerotree image and video coding algorithm that has a significant improvement in the coding memory requirement and the coding complexity, which gives two advantages over other zerotree algorithms. The first advantage is that the proposed algorithm is more suitable for real-time software implementation, as a real-time software codec requires a fast and low complexity coding process, as well as a low memory

overhead. The second advantage is that the proposed algorithm gives a lower cost, lower power consumption hardware implementation than other zerotree algorithms. The reduction in implementation cost is the direct result of the reduction in the coding memory requirement and coding complexity, and the reduction in power consumption is the indirect result of the reduction in the coding memory requirement and coding complexity. The power consumption is proportional to the number of hardware operations [4, 5, 6, 7]. Therefore, a smaller number of operations, as a result of the reduction of coding memory requirement and the coding complexity, will reduce the power consumption.

This proposed low memory, low complexity zerotree coding algorithm was firstly derived for color image coding, and later extended to color video coding.

1.5 Thesis Outline

This thesis is organized as follows.

Chapter 2 gives a review of the human visual system, including the visual pathway, the visual characteristics, and color vision. From the review we will be able to understand how our visual system processes the visual signals and how the visual signals are represented, so that we can mimic those processes and signal representation in visual data coding. Also, by understanding the characteristics of our visual system, we will be able to know what types of visual signals are more important to our visual system, so that we can derive a better rate-distortion optimization scheme to improve compression results.

Chapter 3 discusses the color space transform, wavelet transform, and entropy coding. Their functions are to reduce visual data correlations for obtaining a data representation that has a possibly small amount of visual redundancy. The color space transform and wavelet transform are closely related to human visual system characteristics, as they process the visual data in a way similar to our visual system, so that we can make use of human visual characteristics in visual data compression. On the other hand, entropy coding is based on information theory and is applied to the compressed bit-stream for obtaining a more compact data representation by reducing the probability redundancy. Those processes are essential for improving compression performance.

Chapter 4 reviews general visual data compression schemes and the compression quality assessment methods. The discussion of the visual data compression schemes provides a general classification of the available options and gives an idea of how the algorithm proposed in this thesis compares with those compression schemes. The quality assessment is important not only

to let us know the performance of each compression scheme, but also to provide information on how to improve compression performance.

Chapter 5 discusses zerotree coding theory in detail. Following the discussion of the theory, the key zerotree image and video coding algorithms that are related to the works of this thesis are reviewed. The drawbacks of those zerotree coding algorithms are also identified in this chapter.

Chapter 6 presents a zerotree image coding algorithm that overcomes the drawbacks of the other zerotree algorithms identified in chapter 5. This algorithm is named the *Listless Zerotree Coding* (LZC) algorithm because it does not use any coding lists, which are normally found in other zerotree algorithms. The theory, coding procedures, and the pseudo-code for the actual software implementation of LZC are also presented in this chapter. The experimental results obtained from the LZC test codec are presented in a variety of parameters and are compared to current benchmark algorithms.

Chapter 7 presents a video compression algorithm which is an extension of the LZC algorithm presented in Chapter 6. Since this video coding algorithm exploits a 3-dimensional wavelet transform for temporal-spatial data decorrelation, it is called *3-dimensional listless zerotree coding* (3DLZC) algorithm. 3DLZC also inherits the features of simple coding complexity and low coding memory requirement from LZC. The experimental results obtained from the 3DLZC test codec are presented and compared to the benchmark algorithms at the end of this chapter.

Chapter 8 suggests two possible extensions that can be implemented into the LZC and 3DLZC algorithm. These extensions include the use of frequency weighting to scale the wavelet subbands to match their corresponding contrast sensitivity functions and the use of video content separation for adding some functionalities to the video coding algorithm.

Chapter 9 summarizes the contributions of this thesis, and makes some concluding remarks, as well as some recommendations for future work.

Chapter 2

The Human Visual System

2.1 Introduction

In order to achieve an image compression with minimal visual distortion, we need to understand what kind of visual information humans cannot perceive, so we can discard it entirely; what kind of visual information is more important than others, so we can allocate more coding bit budget to encode it. Therefore, this chapter reviews the human visual system (HVS), including the visual pathway, color vision and visual system models. This chapter also identifies some fundamental characteristics of the human visual system that are useful from the image compression point of view. This chapter will only give a brief review of the human visual system. For more details on the human visual system, readers are directed to two books written by Hubel [8] and by Wandell [9].

2.2 The Visual Pathway

This section reviews how visual information reaches the brain and how the visual system processes and interprets visual information.

The visual pathway consists of three stages: *retina*, *lateral geniculate nucleus* (LGN) and *visual cortex*. Figure 2.1 illustrates those three stages of the visual pathway and the types of information carried by the pathway. In the figure, green ellipses represent receptors in the retina and yellow circles represent visual neurons. This figure only shows the pathway in one side of the brain.

The retina, which initiates the vision, is a thin layer of neural tissue in the eye. It contains three main types of cells, namely, *receptors*, *bipolar cells* and *ganglion cells*. In Figure 2.1,

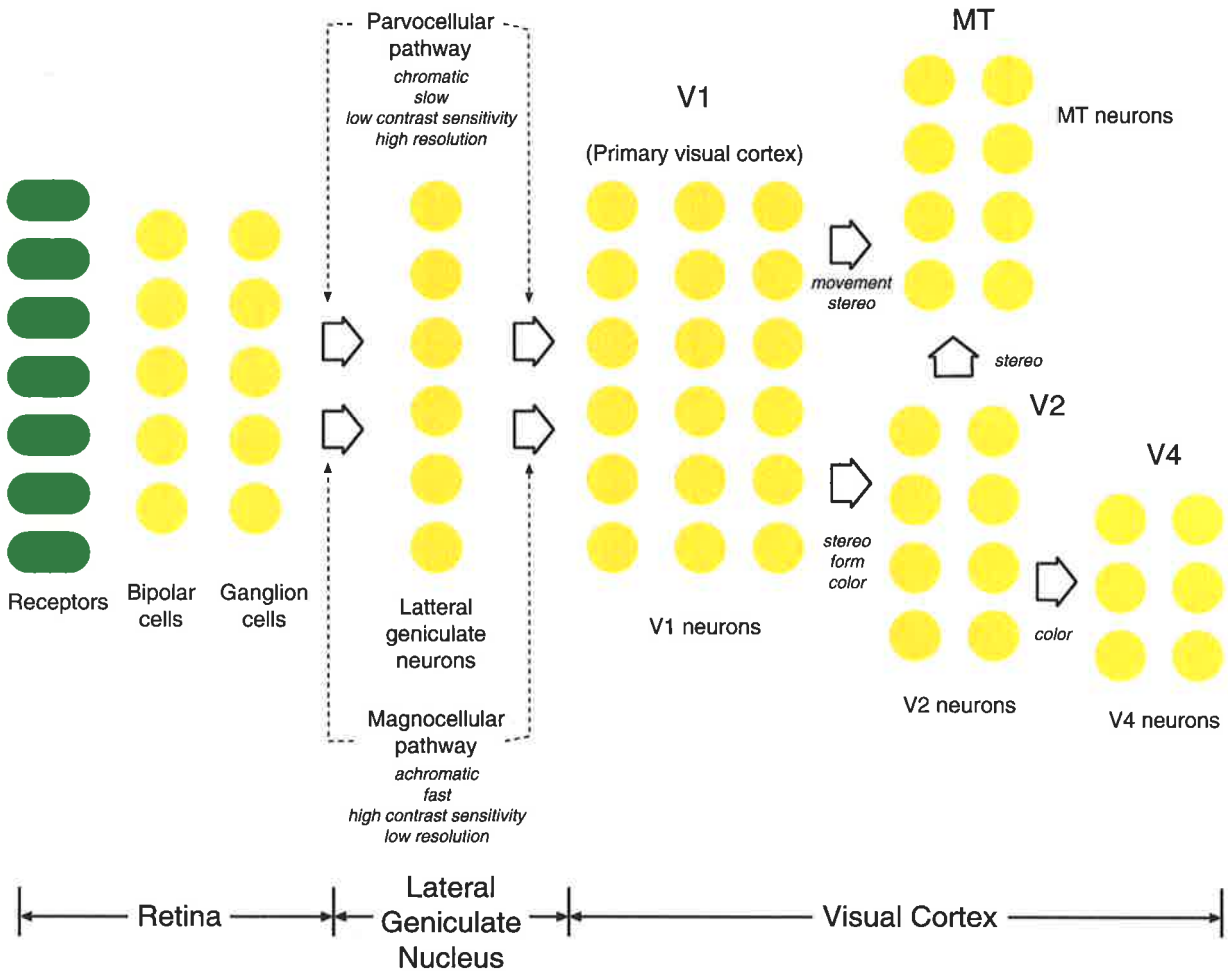


Figure 2.1: The visual pathway

receptors are colored green and the other two types of retinal cells are colored yellow. The function of receptors is to translate light energy into nerve signals. Receptors are connected by bipolar cells to ganglion cells. Ganglion cells receive signals from a number of receptors and transport the nerve signals via their axons, long and thing neuron extensions from cell bodies, to the LGN. The axons of ganglion cells are bundled and leave the eye through the optic nerves.

The receptor to ganglion cell ratio is about 125:1, which implies a signal compression occurring along the visual pathway [9]. This compression is achieved by replacing photographic image information with its shape, color and motion information, which is then encoded by *midget ganglion cells* and *parasol ganglion cells*. The axons of midget ganglion cells form the *parvocellular pathway*, while the axons of parasol ganglion cells form the *magnocellular pathway*. The parvocellular pathway carries chromatic, slow, low contrast and high resolution signals which are important for conscious perception; whereas, the magnocellular pathway carries achromatic, fast, high contrast and low resolution signals which are important for reflex

action [10].

The LGN is the second stage of the visual pathway. Signals carried by the parvocellular pathway arrive at the parvocellular layers of the LGN, whereas signals carried by the magnocellular pathway arrive at the magnocellular layers of the LGN. The function of the LGN is not well understood, but an LGN in each side of the brain receives signals from both retinas and transports these signals to the visual cortex at the same side of the brain.

The visual pathways remain segregated until they arrive at the visual cortex, the third stage of the visual pathway. Nerve signals in both magnocellular and parvocellular pathways are then processed and interpreted by the visual cortex. The visual cortex is divided into area V1 (the primary visual cortex), area V2, area V4, and area MT. Neurons in these cortical areas are tuned to specific signal properties such as orientation, form, color, spatio-temporal frequency, stereo information, or motion. Therefore, these cortical areas process and interpret signals that their neurons are tuned to.

Area V1 is the most important area of the visual cortex. It processes and interprets signals from both magnocellular and parvocellular pathways that enable us to *see* things. Loss of area V1 will result in a total loss of our vision. The functions of the other cortical areas are as follows. Area V2 processes and interprets color, form and stereo signals from area V1; area V4 receives color signals from area V2 for further color interpretation; area MT interprets movement signals from area V1 and stereo signals from both area V1 and V2.

In short, the visual information decomposition along the visual pathway and the tuning of cortical neurons have inspired algorithms that decompose images into different channels in a similar fashion. Multichannel vision models and multichannel image decomposition algorithms that simulate the visual pathway and cortical areas in processing visual information will be discussed later in Section 2.4.

2.3 Fundamental Properties of the HVS

2.3.1 Luminance Sensitivity

The *brightness* of an object is defined as the perceived luminance by the human visual system. However, the perception of brightness is known to be a nonlinear function of the light incident on the eye [11](ch. 2). The brightness of an object depends on the luminance of the surrounding objects. In other words, two objects with the same *luminance* or *intensity* but different surrounding objects could have different perceived brightness. This phenomenon is called *simultaneous*

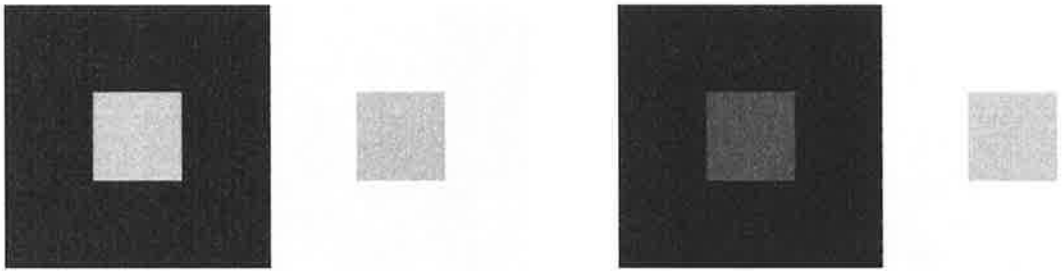


Figure 2.2: Simultaneous Contrast: (a) small squares with the same luminance but different perceived brightness (b) small squares with different luminance but similar perceived brightness

contrast.

Figure 2.2 illustrates the *simultaneous contrast* phenomenon with two examples. In Figure 2.2(a), two small squares have the same luminance but different perceived brightness. Due to different surrounding luminance, the small square on the right appears darker than the one on the left. On the other hand, in Figure 2.2(b), two small squares have quite different luminance, but with the surrounding luminance they appear to have similar brightness. The reason is that although the human vision perception can adapt to an enormous range of light intensity, of the order of 10^{10}, it cannot simultaneously operate over such a range. Instead, it can only discriminate a comparatively small range of light intensity levels simultaneously, due to its brightness sensitivity adaption. That is, the human visual system is sensitive to luminance contrast rather than the absolute luminance levels [12](ch. 3).

Weber's Law characterizes the ability of human visual system to discriminate between changes in brightness at a given brightness adaption level. Suppose that the surrounding luminance is L_s, the current brightness adaption level, and the object luminance is L_o. If L_o is just noticeably different from L_s then their ratio is

$$\frac{|L_s - L_o|}{L_s} = \frac{\triangle L}{L_s} \approx \triangle(logL) = \triangle c \tag{2.1}$$

where $\triangle L = |L_s - L_o|$ and $\triangle c$ is a constant. The constant $\triangle c$ varies depending on the brightness discrimination of the viewer. If the value of $\triangle c$ is *small*, then the viewer is said to have *good* brightness discrimination. In contrast, if the value of $\triangle c$ is *large*, then the viewer has *poor* brightness discrimination. The value of $\triangle c$ has been found to be about 0.02 for average viewers [12](ch. 3). This value implies that at least 50 levels of brightness on a scale of 0 to 1 are needed for representing the gray-scale digital image. Moreover, the logarithmic sensitivity to luminance causes our visual system to be more sensitive to distortion in the dark areas than in the bright areas.

2.3.2 Frequency Sensitivity

The perceived brightness not only changes with the surrounding luminance but also changes with the spatial variation of the luminance levels. For instance, Figure 2.3(a) shows stripes with increasing luminance level from the left to the right. Although each stripe has a constant

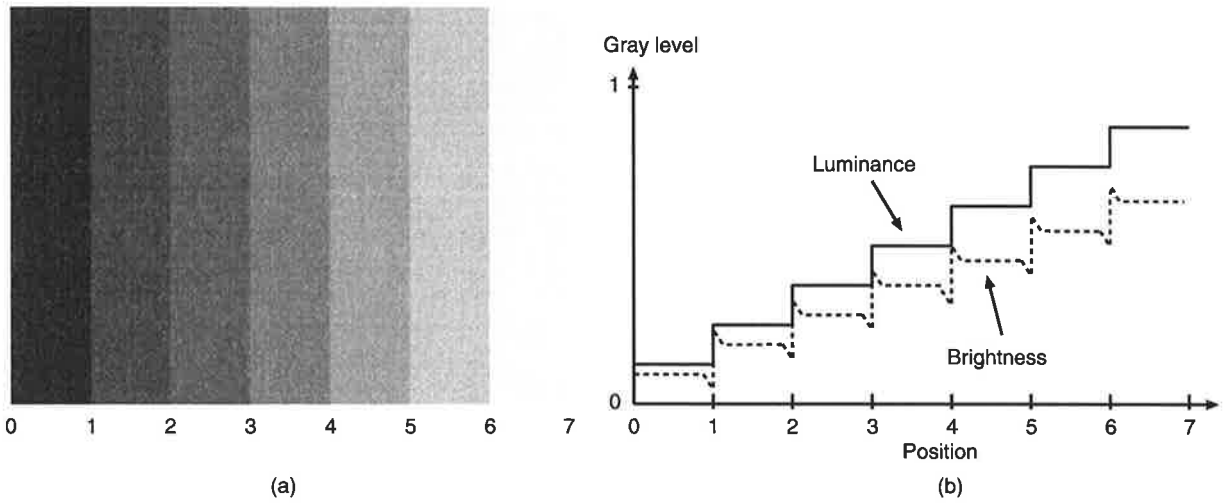


Figure 2.3: Mach band effect: (a) stripes with increasing luminance levels from the left to the right (b) luminance levels of the stripes versus perceived brightness

luminance level, the perceived brightness is not uniform along the width of the stripe. At each boundary transition, the perceived brightness is darkened on the darker stripe side but brightened on the brighter stripe side. Figure 2.3(b) shows the relative luminance level of each stripe versus the perceived brightness. The phenomenon of undershooting or overshooting the perceived brightness by the human visual system is called the *Mach band effect*, named after Ernst Mach, who first described this phenomenon in 1865 [11](p.28).

The Mach band effect implies that the impulse response of the human visual system in the spatial domain is band pass in nature [12](p.54). The frequency response of the human vision system is determined by the threshold *Modulation transfer function* (MTF), which can be measured directly by viewing sinusoid gratings of varying spatial frequencies and contrasts — that is, by measuring the contrast needed for a viewer to detect sinusoid gratings of various frequencies. Figure 2.4(a) shows an example of sinusoid gratings at 3 cycles per degree (cpd) of unity visual angle. The visual angle, α, is given by

$$\alpha = 2tan^{-1}\frac{W}{2D} \quad (degree) \tag{2.2}$$

where W is the width of the gratings and the D is the viewing distance. The Michelson contrast

is generally used to define the contrast of sinusoid gratings [13], and is given by

$$C_M = \frac{L_{max} - L_{min}}{L_{max} + L_{min}} \tag{2.3}$$

where L_{max} and L_{min} are the maximum and minimum luminance level of the sine wave as indicated in Figure 2.4(b).

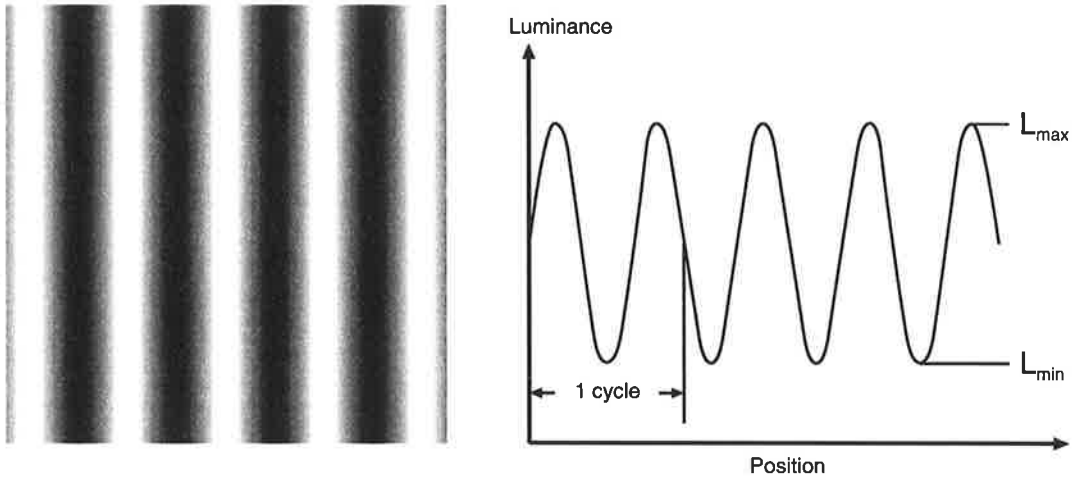


Figure 2.4: (a) Sinusoid gratings at 3 cpd of an unity visual angle (b) luminance of sinusoid gratings

The frequency response of the human visual system is more often represented by the *contrast sensitivity function* (CSF), which is the inverse of MTF. CSF varies with the viewer and the viewing conditions, such as the types of stimuli, display luminance and resolution, viewing distance and angle, and surrounding luminance. Generally the peak of CSF lies between 2 to 10 cpd.

Figure 2.5 shows a typical CSF for display luminance at 100 cd/m^2 and display width of 4 $degrees$. In this case, the peak sensitivity occurs at 8 cpd. This CSF was constructed using the equation presented by Lubin in [14] and is given by

$$CSF(v) = \frac{1}{MTF(v)} = av\, exp(-bv)\sqrt{1 + c\, exp(bv)}, \tag{2.4}$$

where

$$a = \frac{540(1 + 0.7/L)^{0.2}}{1 + 12/[w(1 + v/3)^2]}, \tag{2.5}$$

$$b = 0.3(1 + 100/L)^{0.15}, \tag{2.6}$$

$$c = 0.6, \tag{2.7}$$

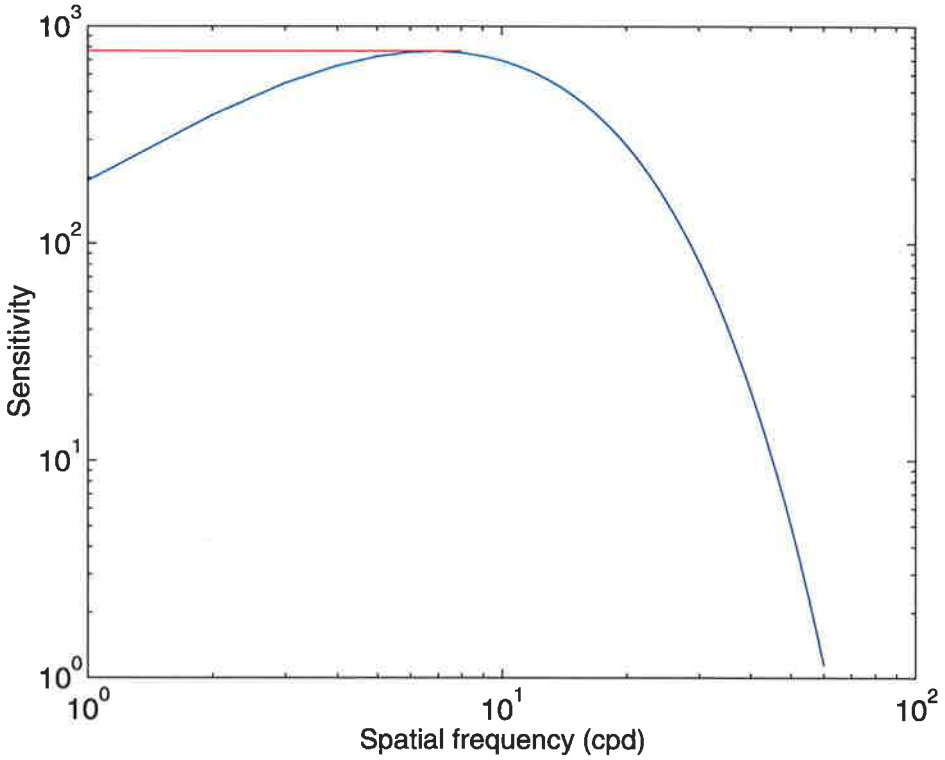


Figure 2.5: A typical CSF of the human visual system

and L and w are display luminance in cd/m^2 and display width in degrees, respectively.

The CSF reflects two striking features of the human vision system [9](p.202). First, the human visual system is not sensitive to high spatial frequency patterns. This is because the optical blurring of the lens reduces the contrast of high spatial frequency patterns, and the retinal ganglion cells with center-surround receptive fields are less sensitive to high spatial frequency patterns. In addition, the human eye can not see patterns with spatial frequency higher than 60 cpd, due to the limited number of photoreceptors in the eye. Second, surprisingly, the contrast sensitivity of human visual system also decreases in viewing low spatial frequency patterns. This may be due to neural factors.

The findings of the CSF measurement suggest that high frequency information in images is less important to the human visual system. Therefore, high frequency information can be quantized coarsely to achieve higher compression. However, although the human visual system is also less sensitive to the low frequency patterns, the low frequency information in images cannot be quantized coarsely. The reason is that the spatial frequency of patterns is proportional to the viewing distance. That is, a low frequency pattern at a close viewing distance will become a high frequency pattern at a further distance. According to Equation (2.2), doubling the

viewing distance will double the spatial frequency, as the visual angle is approximately halved. Therefore, as a precaution, the CSF that is applied for quantizing low spatial frequency patterns should have the curve indicated by the red line in Figure 2.5.

2.3.3 Orientation Sensitivity

Following the gravitational direction, most of our surroundings comprise of horizontal and vertical structures that create a global reference frame, where horizontal and vertical orientations are the axes. Therefore, during the development of the human visual system, the brain is psychologically optimized to perceive horizontal and vertical stimuli. Consequently, the human visual system shows greater sensitivity to horizontal and vertical orientations than to oblique orientations (45°/135°). This property is the so called *oblique effect*, which was observed by Appelle [15].

Furthermore, the *oblique effect* can also be explained from the findings of neuro-science studies. Cortical neurons of the *primary visual cortex* (or area *V1*), have an important property called *orientation selectivity*, which means the cortical neurons only respond to stimuli of their preferred orientations [8](p.115), [9](p.167). The studies of single-neuron electrophysiology [16, 17] and optical imaging [18] find that more cortical neurons are tuned to response stimuli in horizontal and vertical orientations than in oblique orientations. This finding is confirmed by behavioral measurements of the human visual system [19].

Because of the *oblique effect*, the contrast sensitivity for patterns in oblique orientations will be lower than those with the same spatial frequency but in horizontal and vertical directions. That is, patterns in the oblique orientations have higher threshold MTFs. Therefore, oblique information in the image is less important to the human visual system and can be quantized more coarsely than horizontal and vertical information.

2.3.4 Signal Content Sensitivity

Signal content sensitivity of our vision is affected by two visual phenomena, namely, *masking* and *facilitation*. Masking is an effect whereby the presence of one signal (the masker) reduces the visibility of another signal (the target), whereas facilitation refers to an increase of the visibility of one signal due to the presence of another [9](p.219). Masking effects can be classified into two categories: *contrast (pattern) masking* and *texture masking* [20, 21, 22]. Contrast Masking happens when a weak (low contrast) signal is masked by a strong (high contrast) signal that is at the same spatial location. Texture masking occurs when a signal is hidden by a

background signal (the masker), if the background signal is so busy and irregular that we have difficulty in finding the signal.

Legge and Foley [23], and Solomon et al. [24] model contrast masking by superimposing two sinusoidal signals and estimating the conditions when masking occurs, but they only model the masking between two signals having energy in the same channel (ie. same spatial frequency and orientation), so it is called the *intra-channel masking* model. Later, the masking model was modified by Foley [25], Teo and Heeger [26], and Watson and Solomon [27] by measuring masking between sinusoidal signals having energy in different channels (ie. different spatial frequencies and orientations); therefore, it is called the *inter-channel masking* model. These models estimate the threshold level in detecting the contrast of a signal, in the presence of a masker of a different spatial frequency, orientation, contrast and phase.

Within the framework of image compression, we are more interested in the ability of an image content to mask distortion or quantization error. That is, the original image is the masker and the distortion is the target signal. However, the contrast masking model is too simple to be useful for image compression, since it only models the masking between two sinusoidal signals and cannot measure the effect of texture masking. Even modeling masking by taking account of noise patterns, ie. noise masking [28, 29], still cannot represent the actual masking that occurs in a natural scenery image.

The masking threshold depends on the familiarity of the image to the viewer. Viewers generally have some *a priori* knowledge about what simple image contents, such as edges or human faces, look like. Therefore, the degree of masking is normally small on these simple and easily learned contents. On the other hand, texture regions in an image have a significant degree of masking, because the texture regions are less predictable and hard to learn. Nevertheless, if viewers are given enough time to learn image contents, they will become familiar with those texture regions, and the degree of masking on texture regions will be reduced [30].

Hence, a more complex masking model is essential to estimate the degree of ability of natural scenery images to mask real distortion or quantization errors. Blackwell [31], Burgess et al. [32], and Eckstein et al. [33] present masking measurement with non-deterministic noise patterns that simulate realistic distortion. Watson et al. [34] and Nadenau et al. [22] perform some masking experiments with more complex backgrounds from natural scenery images and more closely measure texture masking in real applications like image compression. Their masking models are used for a quantitative measurement of the image quality that is similar to the quality perceived by human observers.

Texture masking influences the error tolerance of the viewer toward the compressed image, and explains why similar quantization errors are bothersome in smooth regions but are barely

visible in texture regions. For maximum masking to occur, both the masker and the target must be at about the same spatial location, have about the same spatial frequency, and have about the same orientation. Quantization errors generally are high spatial frequency signals. If these errors are introduced to low spatial frequency regions where there is no high frequency component, then these errors will be *facilitated* by the low frequency regions. This implies that masking and frequency sensitivity of the human visual system are closely related. Both suggest that the human vision is less sensitive to noise at high spatial frequencies, as in the busy part of an image. Hence, the texture regions of an image can be quantized more coarsely than the smooth regions [35].

Figure 2.6 illustrates the three masking effects that have been discussed: contrast masking, noise masking, and texture masking. The images in the left column are the targets; the images in the middle column are the masks; and the images in the right column are the masking results.

2.3.5 Temporal Sensitivity

Temporal sensitivity is related to video processing and compression. It involves three issues: *critical fusion frequency*, *temporal contrast sensitivity*, and *temporal masking*.

Critical Fusion Frequency

Critical fusion frequency (CFF) is the transition frequency of a flickering light, above which the flickering light will appears as a steady light of the same average intensity [36]. Our CFF varies depending on the viewing conditions and stimulus, but generally this frequency is about 50 to 60 Hz [12]. CFF is important to determine the sampling and display rates of interlaced video. The rates are 50 or 60 Hz depending on the power-line frequency of the application location. Also, the refresh rate of a computer monitor should be higher than 60 fps to avoid any flicker perception.

Temporal Contrast Sensitivity

Similar to spatial contrast sensitivity, the temporal contrast sensitivity of our vision is also described by the temporal CSF. The temporal CSF has a band-pass or low-pass filter shape like spatial CSF. The temporal CSF also varies depending on the viewing conditions, stimulus types, and, of course, viewers. Generally our temporal CSF have peaks at about 15 HZ and cut-offs at about 60 Hz. The cut-off frequency of 60 Hz explains why CFF is about 60 Hz. Temporal

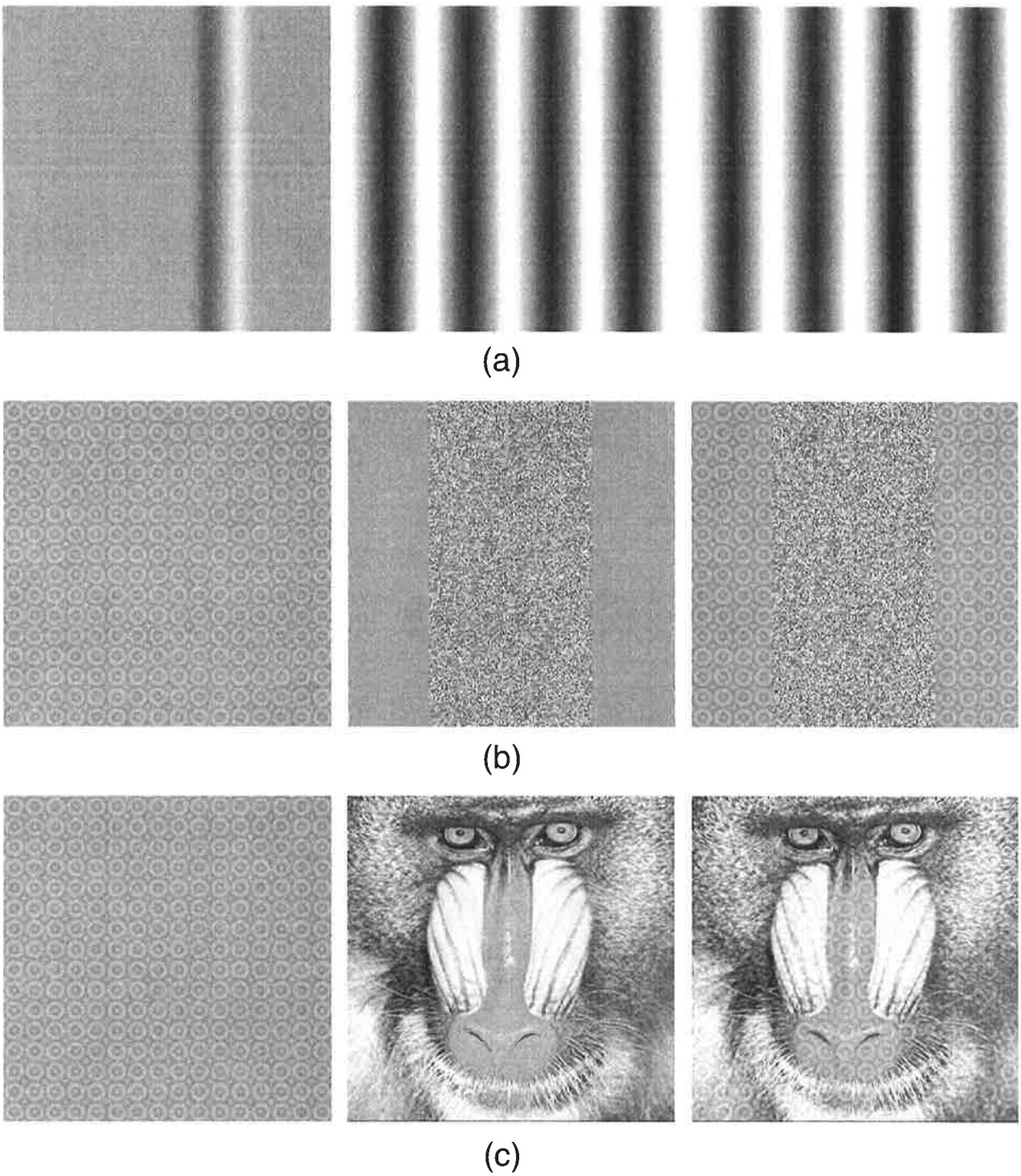


Figure 2.6: Masking effects: (a) contrast masking (b) noise masking (c) texture masking

CSFs often are combined with spatial CSFs to model our visual spatio-temporal contrast sensitivities [37, 38]. However, to simplify computations and representation the spatio-temporal CSFs are modeled as combinations of functions separable in space and time [39, 40, 41, 42]. The spatio-temporal CSFs are commonly used in vision models for video.

Temporal Masking

Temporal masking is very important for video coding. High temporal contrast sensitivity at around 15 Hz indicates that at around this frequency our visual system has less temporal masking ability and is very sensitive to distortions. However, the masking effect is complicated and heavily dependent on whether or not the object is being tracked by the eye [43]. If the object is moving too fast or in irregular and unpredictable fashion, then the viewer loses perceived spatial resolution and is relatively insensitive to distortion of the object. The degradation of our visual resolution also happens when there is a sudden *scene change* in video. At the point when the video scene changes, the perceived spatial resolution on the new scene decreases significantly to about one-tenth of the peak contrast sensitivity threshold for a few hundred milliseconds [44]. Interestingly, the temporal masking caused by scene changes can also occurs on the old scene just before the change. This *backward temporal masking* is due to the variation in the latency of the neural signals in the visual system as a function of their intensity [45]. Because of the modeling complexity, the temporal masking effects have been less applied to video coding than their spatial counterparts.

2.4 Multichannel Representation of the HVS

Early vision models adopted the single-channel (or single-resolution) theory that regarded the human visual system as a single spatial filter [29, 46, 47, 48]. The characteristics of single-channel models are defined by the contrast sensitivity function. Although single-channel models provide an insight into some simple visual phenomena, they fail to explain more complex phenomena like pattern adaption and masking. To explain these phenomena, more complex vision models are required.

Empirical data from the measurement of *receptive fields* (RF) of the primary visual cortex neurons, the so-called *simple cells*, suggest multichannel (or multiresolution) vision models [49, 50, 51]. Measurements show that receptive fields of the simple cells have a number of interesting properties. First, the receptive field is local in space and covers only a small region of the entire spatial visual field [52]. This shows that the receptive fields have a linear spatial summation property, as the entire visual field is the summation of the receptive fields [53].

Second, the receptive field responds to a certain range of spatial frequencies and is local in two-dimension spatial frequencies [50, 54, 55]. This implies that the frequency response of the receptive fields is band-limited. Also, there must be different types of receptive fields to cover the entire spectral visual field. Third, the receptive field is only sensitive to a certain orientation [17, 49]. This means the receptive fields have the orientation selectivity property and there must be a variety of receptive fields to respond to different orientations. Figure 2.7 illustrates the receptive fields of simple cells which are sensitive to four different orientations.

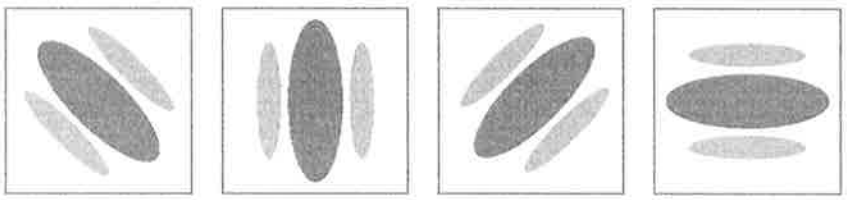


Figure 2.7: Receptive fields of cortical simple cells

The properties of the simple cell receptive fields allow the human visual system to be modeled by multi-channel models. That is, the visual pathways consist of a combination of different convolution kernels that are tuned to a number of selective spatial frequencies and orientations. For the achromatic visual pathways, the spatial frequency bandwidth is approximately 1 to 2 octaves and the orientation bandwidth is about 20 to 60 degrees [17, 54, 56]. Those results coincidentally match the statistic of natural images [57]. Wilson and Regan [58] suggest that an image stimulus can be decomposed into 48 components, the so-called *neural images*, including six spatial frequency channels and eight orientations.

Watson [59], Daly [60] and Lubin [14] have proposed image transforms that are analogous to the multichannel decompositions of visual pathways. Those transforms are called *cortex transforms*. Figure 2.8(a), (b) and (c) show the cortex transforms that are proposed by Watson [59], Daly [60] and Lubin [14], respectively. The cortex transforms resemble receptive fields which decompose the spectral visual field into selective frequency resolutions and orientations. The frequency bandwidths are 1 octave for all three transforms, but the orientation bandwidths are 30° for Daly's and 45° for both Watson's and Lubin's. Figure 2.8(a) illustrates the multichannel decomposition of the spectral visual field by Watson's cortex transform. The spectral visual field is decomposed into six frequency resolutions, as indicated by the frequency *band* number, and four orientations, as marked by the *fan* number.

In short, multichannel vision models provide a better understanding of how the visual system processes information. The multichannel models also help us to identify useful visual system characteristics for image processing. In addition, cortex transforms provide a convenient means to model the multichannel decomposition of the visual system, so that the visual information

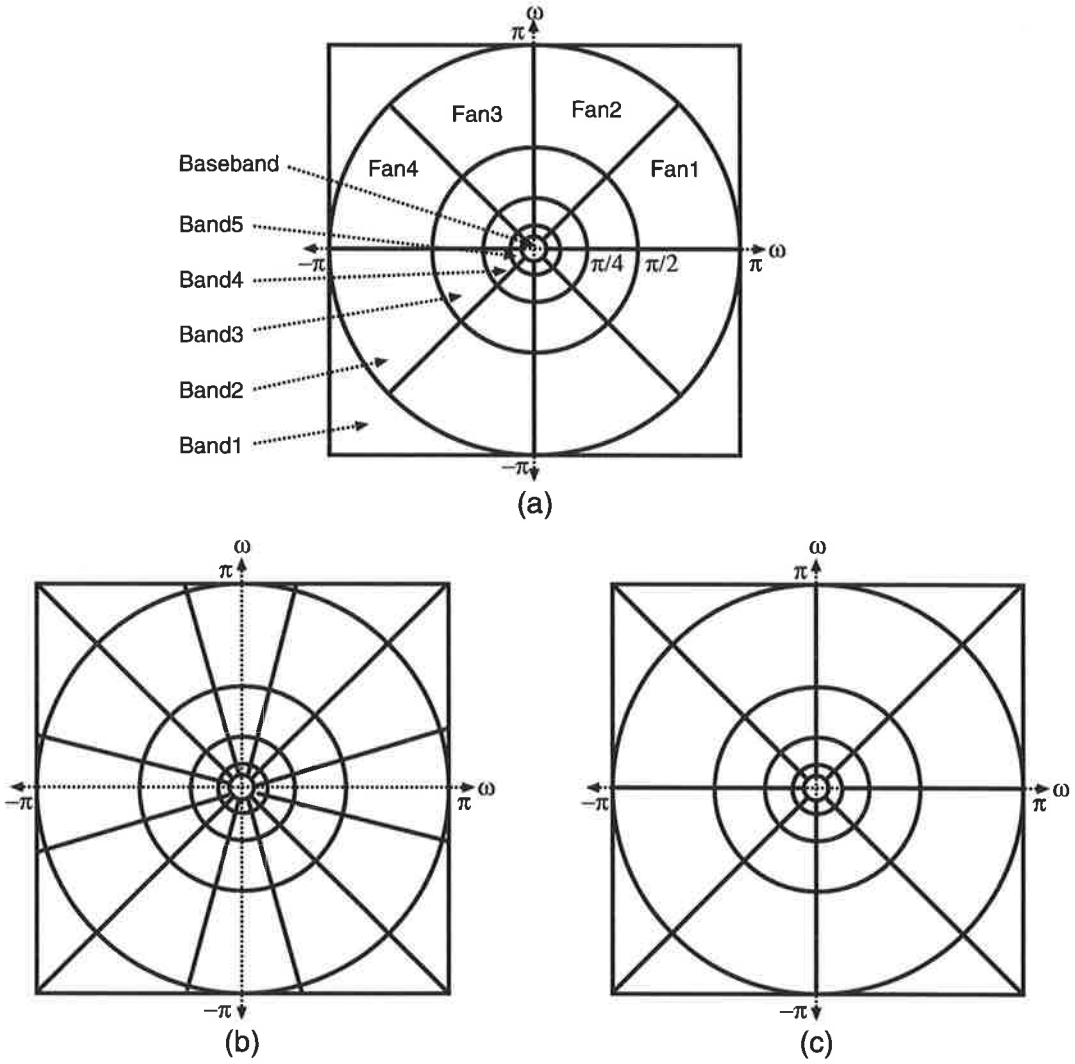


Figure 2.8: Cortex transforms in the frequency domain (a) by Watson [59] (b) by Daly [60] (c) by Lubin [14]

can be processed in a way that can take advantage of the visual system characteristics.

2.5 Color Vision

The definitions for color vision from psychology and physics are as follows. Color is a subjective sensation of human perception [61] and objects do not have color in nature. Without light and an observer, color does not exist, because color is a sensation conveyed through the medium of energy in the form of light radiations within the visible spectrum. As Newton mentioned in his *Optiks*: "*The rays are not colored. In them there is nothing else than a power to stir up a sensation of this or that color*". However, in order to identify color information

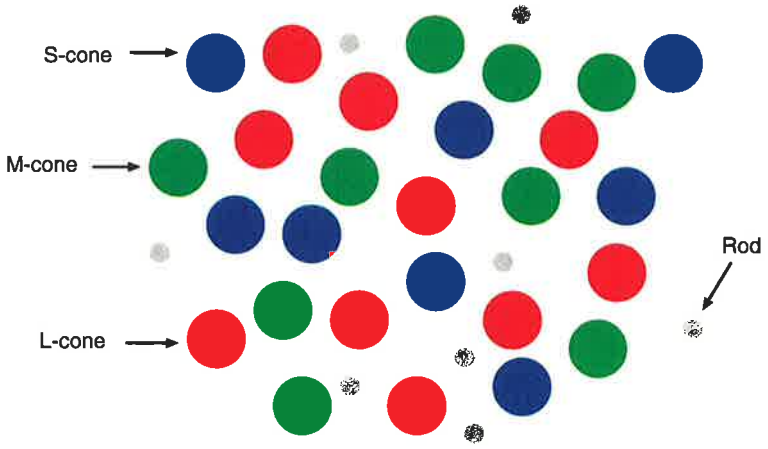


Figure 2.9: An illustration of retinal receptor mosaic

that is redundant to the human visual system, we need to look for color vision definitions from psychophysics and physiology. Psychophysics examines the capabilities of the visual system in processing color information, and physiology examines how color vision works.

2.5.1 Trichromatic Vision vs. Opponent Color Vision

As mentioned in Section 2.2, visual receptors in the retina translate light energy into nerve signals, and optical nerves transport those signals along the visual pathway to the visual cortex for color interpretation. There are two types of visual receptors (or *photoreceptors*) in our retina, *rods* and *cones*, which respond to brightness and color, respectively. The distribution of rods and cones varies across the retina. The center of visual field, the so-called *fovea*, contains only cones and has the highest visual acuity. The visible spectrum for these two types of visual receptors ranges from $380nm$ to $780nm$ [62](p.6). Rods are very sensitive visual receptors: they can respond to low, *scoptic* light levels like star light or dim light. Under such condition, cones are not sensitive enough to respond, so the vision completely lacks color. On the other hand, cones initiate color vision under high, *photopic* light levels, under which images are usually viewed. Rods are more or less saturated under photopic light levels. For this reason and also because rods do not exist in the fovea, their responses are generally neglected for image processing and compression.

There are three types of cones in the retina, namely, L-cones, M-cones, and S-cones with peak sensitivity corresponding to the long-, middle-, and short-wavelengths, respectively [9] (ch.3). The spectrum sensitivity of L-cones, M-cones, and S-cones also reflects the *spectral power distribution* (SPD) of *red* (R), *green* (G), and *blue* (B) light, respectively. Therefore, L-cones, M-cones, and S-cones are sometime called R-, G-, and B-cones. Figure 2.9 illustrates

the retinal receptor mosaic with red, green, and blue circles representing L-, M-, and S-cones, and gray circles representing rods. The exact sensitivity spectrum of cones varies, however, depending on the measuring methods and the human observers [63, 64, 65]. Nevertheless, the light wavelength coding of these three types of cones provides weighted summation over the different sensitivity spectra. This suggests that three values should be sufficient to reproduce human color distinction capabilities. This matches Young-Helmholtz *trichromacy theory*, which states that any color could be obtained by mixtures of light of three colors, red, green and blue. An observer being able to discriminate three primary lights and perceive correct colors is said to have *trichromatic vision* [66] (p.69). Trichromacy theory is important for modern color applications, like image processing and compression, in obtaining compact representation of colors in the visible spectrum.

Interestingly, the trichromatic representation of our color vision only happens in the retina. After the retina, color information is encoded by the so-called *opponent color representations* at the subsequent stages of visual pathway [8](ch.8), [9](ch.9), [67]. Opponent color representation was first described by Hearing (1878). He proposed that color information in the eye, brain, or both are represented by three *opponent processes*, one for blue-yellow (B-Y) sensations, one for red-green (R-G), and the third for black-white (B-W). Hearing's opponent-color theory was quantitatively proved by the *hue-cancellation experiment* conducted by Jameson and Hurvich in the mid-1950s [68, 69]. They confirmed that color information in the visual pathway is represented by the differences of cone responses, rather than by the cone responses directly. Opponent-color theory explains how the visual system transports color information with limited visual channel capacity, as there are fewer optic nerves than visual receptors and each optic nerve has a limited dynamic range [70]. That is, the blue-yellow and red-green channels, which are analogous to the parvocellular pathways, carry visual signals representing the difference of cone responses; while the black-white channel, which is analogous to the magnocellular pathway, carries visual signals representing the weighted sum of cone responses.

Figure 2.10(a) shows the opponent color processing of two center-surround receptive fields: the left one which has a red-ON center and green-OFF surround is responsible for red-green opponent process; the right one which has a blue-ON center and yellow-OFF surround is responsible for blue-yellow opponent process. The schematic illustration of opponent color coding of the visual pathway is shown in Figure 2.10(b). Representing color informations by two chromatic channels and one luminance channel not only significantly decreases the redundancy between the cone responses [71], but also statistically follows the principal components of natural sceneries [72].

In short, trichromacy theory enables colors to be represented by a three-dimensional vector space, which makes computations with color values possible. This has led to the formation of

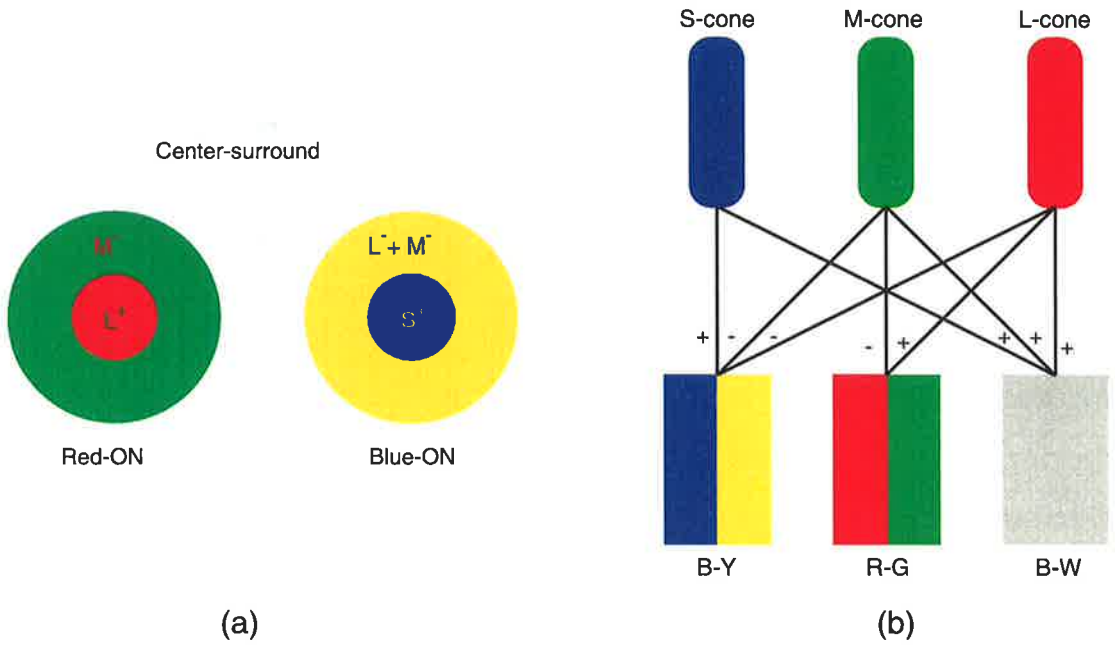


Figure 2.10: (a) Opponent color processes of two center-surround receptive fields: one with red-ON center and one with blue-ON center (b) schematic illustration of opponent color coding of the visual pathway

several widely used color standards (or color spaces) like XYZ and RGB [73]. Furthermore, opponent processes in the visual pathway illustrate an efficient color representation with less color redundancy. Based on *null responses* (hue-cancellations) of color-opponent visual neurons, Derrington *et al.* propose a color space which has two chromatic components and one luminance component [74]. This coding is later exploited in several color spaces, such as YUV and YCbCr, in image processing and compression. Because trichromatic color spaces and opponent color spaces are important to image processing and compression, they will be discussed in more detail in the next chapter.

2.5.2 Color Contrast Sensitivity

As discussed in earlier sections, contrast sensitivity functions (CSFs) are good indicators for identifying image information that is not important and can be quantized coarsely. Therefore, this section briefly discusses CSFs for color information and points out what color information is not important to our visual perception and can be quantized coarsely. Because image compression generally is performed in the opponent color space, we are more interested in the contrast sensitivity functions (CSF) of blue-yellow, red-green and black-white channels, than of red, green and blue channels.

The black-white channel is a luminance channel, so the black-white CSF has the same spatio-temporal characteristics that have been discussed in earlier sections. The luminance channel generally has a higher spatio-temporal CSF than the chromatic channels [35, 75, 76], especially at high spatio-temporal frequencies. Also, the red-green CSF is higher than the blue-yellow CSF. Therefore, at low spatio-temporal frequencies, when blue-yellow, red-green and black-white channels are all sensitive to the visual signals our vision is trichromatic. At moderate spatio-temporal frequencies, the sensitivity of blue-yellow channel declines and our vision becomes dischromatic, as we fail to perceive the blue-yellow variations. At high spatio-temporal frequencies when our ability to perceive red-green variations also diminishes, our vision becomes achromatic, as we perceive all image contrast as black-white modulations of the mean color [9](ch.7). The spatial and temporal frequency sensitivity ranges for chromatic channels is below 12 cpd and 20 Hz, respectively, while the luminance channel has spatial and temporal frequency sensitivity ranges of 0-60 cpd and 0-60 Hz.

For image compression, we need to preserve as much of the luminance information as possible for two reasons. First, because the luminance CSF is the highest among the three channels, it has the lowest masking ability. Second, because the luminance signals are the summation of responses of three cone types, it contains spatial details of the image. Therefore, quantizing luminance information coarsely will cause the reconstructed image to have noticeable distortions. In contrast, having lower CSFs and containing only differences of the cone responses lets chromatic channels have higher masking abilities and contain lower spatial details. Therefore, chromatical signals can be quantized more coarsely to achieve higher compression.

2.6 HVS Model and Visual Insensitivities

More detailed modeling of the human visual system has been attempted previously [77, 78, 79]. This section only tries to provide a simple human visual system model that summarizes the visual properties being discussed in the past few sections. Based on this visual model, this section also gives a brief summary of visual insensitivities that can be exploited for image compression.

A simple human visual system model which accounts for several psychophysical properties is presented as a sequence of processes in Figure 2.11. These processes are summarized as follows:

- *Opponent color processes* transform retinal signals from trichromatic into opponent color representation which is a more compact form for coping with the limited visual channel bandwidth. Also, the opponent color signals are less correlated as they are the differences

Figure 2.11: Block diagram of a typical HVS-model

of cone responses.

- *Multichannel decomposition* is a process that decomposes visual signals into several orientations and frequency resolutions. Multichannel decomposition also results in a more compact visual signal representation. This is not only due to limited bandwidth of the human visual channels but also because the cortical neurons are tuned to respond to their preferred orientations and frequencies.

- *Local contrast adaptation* allows the human visual system to perceive an incredibly wide range of luminance levels. The human visual system simultaneously adapts the local ambient light luminance level and perceives the luminance from objects as brightness (relative luminance).

- *CSF compensation* determines the visual threshold in perceiving the signal contrast at different spatio-temporal frequencies. The human visual system has high contrast sensitivity for low spatial frequency signals but low contrast sensitivity for high spatial frequency signals. Also, the human visual system is sensitive to intermediate temporal frequency signals but less sensitive to high and low temporal frequency signals. These make CSF compensation a spatially low-pass and temporally band-pass process.

- *Masking effects* are related to CSF compensation and enable our visual system to have higher distortion tolerance toward the high spatio-temporal frequency signals. That is, the human visual system has a lower ability to recognize distortions in noisy or highly textured image regions, and sudden changes in video scenes.

Based on the human visual properties, the human visual insensitivities are summarized as follows:

- Color insensitivity means that the human visual system is less sensitive to the distortion in the opponent color signals than the luminance signals. This is because opponent color CSFs of our visual system are lower than the luminance CSF. Also, the blue-yellow channel has a lower CSF than the red-green channel, so the blue-yellow channel has a higher distortion tolerance than the red-green channel.

- Orientation insensitivity means that due to the oblique effect, our visual system is less sensitive to the signals in the oblique orientations than in the vertical and horizontal orientations. This is because psychologically we pay less attention to objects in the oblique orientations, and also because the cortical neurons which are tuned to respond to oblique information have less contrast sensitivity. Therefore, after multichannel decomposition, visual channels in the oblique orientation will have higher distortion tolerance than those in the vertical and horizontal orientations.

- Frequency insensitivity is related to the CSFs of our vision. Having low contrast sensitivity for high frequency signals implies that after multichannel decomposition high frequency visual channels are less sensitive to distortion than low frequency channels. Therefore, high frequency channels can tolerate more errors, while small errors in low frequency channels may cause noticeable perceived distortion to viewers.

- Luminance insensitivity is the result of the local contrast adaptation process (ie. simultaneous contrast) of the human visual system. The human visual system can only perceive a small range of luminance and discriminate a limited number of brightness levels. Therefore, there is no need to have a large number of brightness levels with infinitesimal steps. The result of contrast discrimination experiments suggest that about 50 brightness levels for the scale 0 to 1 will be sufficient. More than that may not be perceivable by viewers and may become redundant.

In short, the properties of the human visual system are very important for image and video compression. The rest of this thesis will discuss how the visual system model and the visual insensitivities can be applied to the proposed listless zerotree image and video coding algorithms.

Chapter 3

Key Elements of Image Compression

3.1 Introduction

Besides the visual insensitivities discussed in the previous chapter, there are three types of data redundancies that can be exploited to achieve better compression results. These redundancies are *color redundancy*, *spatial redundancy*, and *statistical redundancy*.

Color redundancy and spatial redundancy can be reduced by the color space transform and the wavelet transform, respectively. The color space transform and the wavelet transform can also result in a visual data representation that matches the characteristics of the human visual system. Statistical redundancy occurs on the source coded bit-stream and can be reduced by entropy coding.

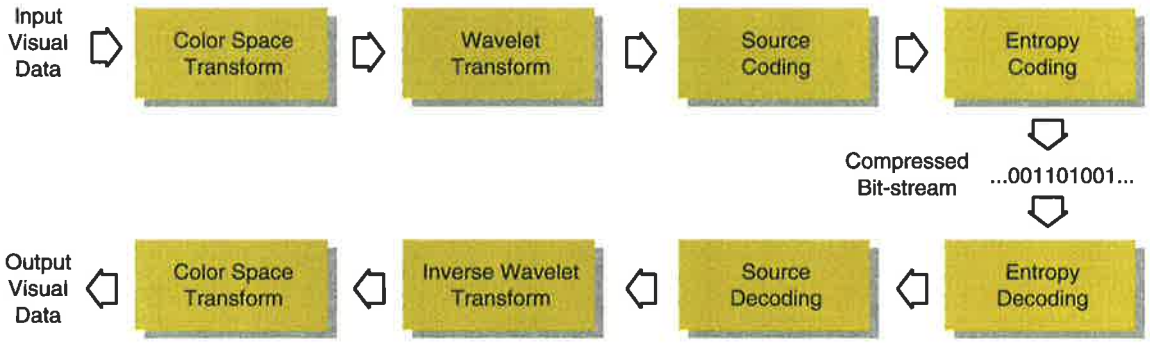


Figure 3.1: A wavelet image compression scheme

Figure 3.1 shows the sequence of applying color space transform, wavelet transform and entropy coding into a wavelet image compression scheme. Although the *core* element of the compression scheme is source coding, the performance of the entire compression scheme is

closely related to how well the data redundancies can be reduced. Therefore, color space transform, wavelet transform, and entropy coding are the *key* elements of a compression scheme, and will be discussed in more detail in this chapter.

3.2 Color Spaces

This section will discuss how the characteristics of the human color vision are exploited in forming different color spaces. Color spaces are used to specify the standard color representation. They can be classified into two categories: *trichromatic color spaces* and *luminance-chrominance color spaces*. Trichromatic color spaces are analogous to our trichromatic vision and produce efficient color spectrum coding, while luminance-chrominance color spaces are analogous to our opponent color vision and produce compact visual signal representation.

3.2.1 Trichromatic Color Space

XYZ Color Space

As discussed in the previous chapter, our vision is trichromatic and in the retina there are three types of cones that respond to long-, middle- and short-wavelengths. Therefore, three values will be sufficient to specify the perceivable color of our vision. However, the color specification is subjective and varies depending on the viewer.

In order to have a standard specification of color, Commission Internationale de l'Eclairage (CIE) introduced the CIE 1931 XYZ color standard in 1931 [80, 73]. The standard was established from the data collected from the *color-matching experiment*, which was based on trichromatic vision theory and Grassmann's additivity law [66] (p.40). The superposition of three primary stimuli from red, green and blue lights was projected on to white wall and was compared to the test color stimulus by the *CIE 1931 standard observer*. The numerical results of these three primary stimuli are called *color-matching functions*, as shown in Figure 3.2(a).

Color-matching functions were used to calculate the CIE tristimulus values, *X, Y, Z*, which represent the the relative amount of the primaries needed to match any pure color by additive color mixture. Stimulus X, Y, and Z encode the spectra corresponding to long-, middle- and short-wavelengths, respectively.

The real colors are then specified by the CIE 1931 (x,y) *chromaticity diagram*, as shown in Figure 3.2(b). The curved line of the shark-fin-like area is called the *spectrum locus*, ranging

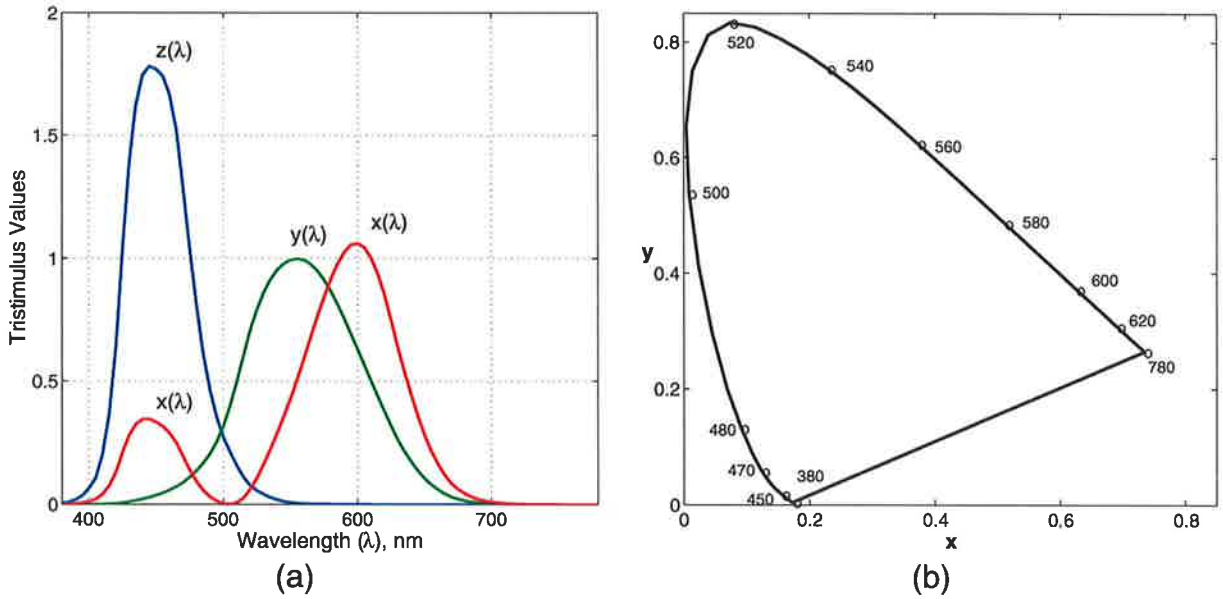


Figure 3.2: (a) CIE 1931 color-matching functions (b) CIE 1931 Chromaticity Diagram

from $380nm$ to $780nm$. The bottom line of the shark-fin-like area is called *pure purple line*, specifying colors from the mixture of red and blue primaries. The area outside the shark-fin-like area represents imaginary colors and has no practical applications.

Although CIE specifies three stimulus values, the chromaticity diagram has only two coordinates, x and y. The is because x and y coordinates are the normalized values of stimulus X and Y and are given by

$$x = \frac{X}{X+Y+Z} \tag{3.1}$$

$$y = \frac{Y}{X+Y+Z}. \tag{3.2}$$

Clearly, two coordinates are sufficient to represent the tristimulus values, as

$$x + y + z = 1. \tag{3.3}$$

The CIE 1931 tristimulus color standard dramatically reduces the amount of numerical data required to represent the entire visible color spectrum. This is one type of compression that is very well understood but seldom mentioned in the image and video compression literature.

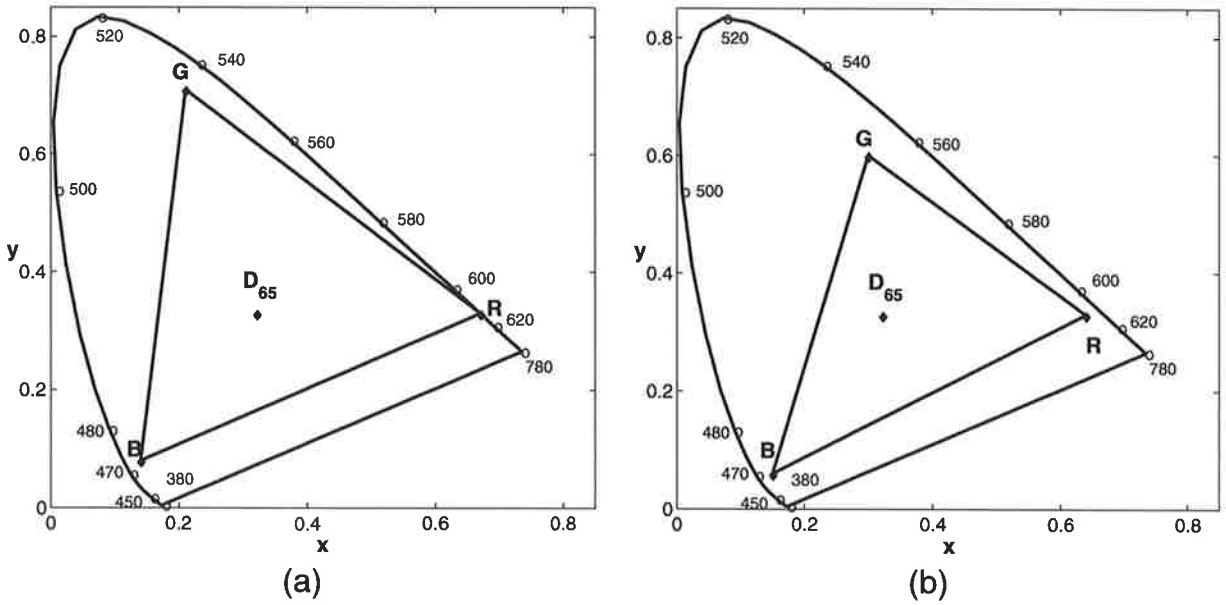


Figure 3.3: RGB gamuts (a) 1953 NTSC (b) HDTV

RGB Color Space

Red, green, and blue are chosen as the primary colors for modern display devices, such as color cathode ray tube (CRT) monitors and TVs. This is not only because we have three types of retinal cones to sense red-, green- and blue-lights, but also because red, green and blue-lights can reproduce the greatest range of colors with a maximum level of brightness.

The red, green, and blue-colors recommended by the U.S. National Television System Committee (NTSC) in 1953 are respectively marked as R, G, and B on the CIE 1931 (x,y) chromaticity diagram in Figure 3.3(a). R, G, and B are the primary colors produced by the RGB phosphors inside CRTs. The triangular area represents colors that can be reproduced by the mixture of RGB primaries and is called the *color gamut*. However, RGB phosphors inside today's CRTs are different from those in 1953. The RGB primaries recommended by the International Telecommunication Union Radiocommunication Sector (ITU-R) [81] for HDTV (High-Definition TV) standards are shown in the Figure 3.3(b). D_{65} inside both NTSC and HDTV color gamuts is called *CIE standard illuminant*, which specifies chromaticity for equal primary signals, ie. reference white. The coordinates of NTSC and HDTV RGB primaries on the CIE 1931 (x,y) chromaticity diagram are summarized in Table 3.1.

Figure 3.4 illustrates the additive characteristic of the RGB color space using the *color cube*. When all RGB primaries are not excited, ie. $(R, G, B) = (0, 0, 0)$, the additive color is *black*, such as shown in Figure 3.4(a). Whereas, when all RGB primaries are at the maximum excita-

	NTSC		HDTV	
	x	y	x	y
Red (R)	0.6700	0.3300	0.6400	0.3300
Green (G)	0.2100	0.7100	0.3000	0.6000
Blue (B)	0.1400	0.0800	0.1500	0.0600
D_{65}	0.3127	0.3290	0.3127	0.3290

Table 3.1: CIE 1931 chromaticity coordinates of NTSC and HDTV RGB primaries

tion levels, ie. $(R, G, B) = (1, 1, 1)$, the additive color is *white*, such as shown in Figure 3.4(a).

Because of the simultaneous contrast phenomenon of our visual system, the excitation range of each primary can be represented as a finite number of discrete levels. For digital image and video applications, the excitation range of each primary is usually divided into 256 levels, from 0 to 255, and represented by an 8-bit (1-byte) number. Therefore, to represent a RGB excitation level will require a 24-bit number, which means there are about 16 million colors that can be produced by the mixture of RGB primaries. A display device that can display 24-bit colors is often said to have 24-bit *color-depth*.

3.2.2 Luminance-Chrominance Color Spaces

Although RGB color space matches our trichromatic vision, it is not an efficient color space for image and video compression. This is because, like our trichromatic vision, RGB color representation has too much information correlation among the three color components, ie. *color redundancy*. Therefore, in order to have a more compact representation for color information, we need color spaces that resemble the opponent color processes in our visual system. That is, we need color spaces that contain RGB color differences, rather than the RGB absolute values. This type of color space may be called *luminance-chrominance color space*.

The most commonly used luminance-chrominance color spaces for image and video applications are YCbCr, YUV, and YIQ. YCbCr is used in both Joint Photographic Experts Group (JPEG) and Moving Picture Expert Group (MPEG) standards; whereas, YUV and YIQ are the standard color spaces for phase alternating line (PAL) TV and NTSC TV systems, respectively. In these three color spaces, CbCr, UV, and IQ are the chromatic components (or chrominances) that contain RGB color differences; while Y is the luminance component that contains the weighted summation of RGB values. Often the chromatic components are called *chromas* and the luminance component is called *luma*.

The Y components of these three color spaces are analogous to stimulus Y of CIE 1931 XYZ

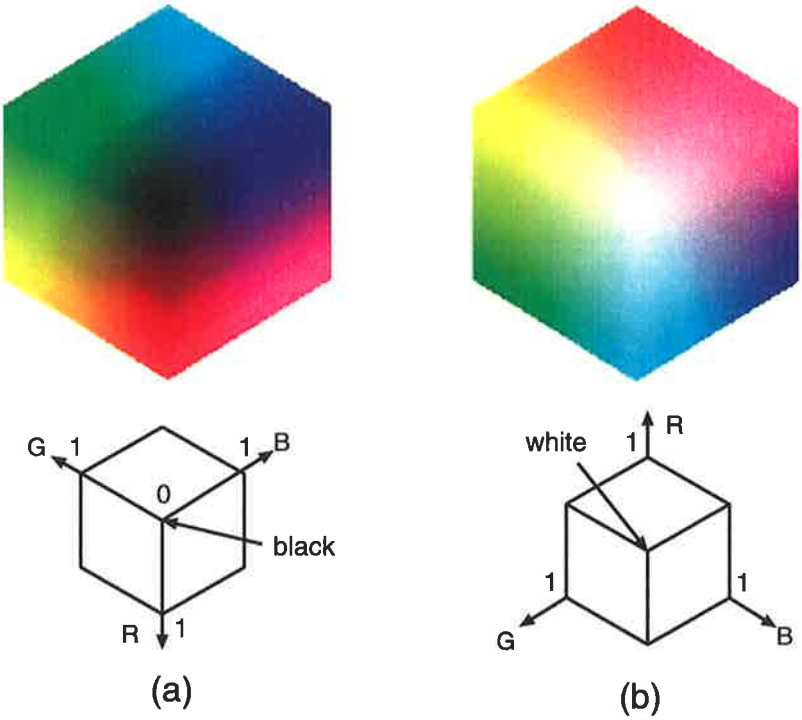


Figure 3.4: RGB Color Cube (a) for black color (R,G,B)=(0,0,0) (b) for white color (R,G,B)=(1,1,1)

color space. Stimulus Y corresponds to middle-wavelength (green) light, which has spectral power density (SPD) spreading over most of the visible spectrum. Therefore, the Y component can well represent the brightness information. However, as obtained from the linear addition of RGB components, the value of the Y component varies depending on the RGB weights specified in different ITU-R Recommendations. In Recommendation ITU-R BT.601 (Rec. 601) [82], the Y component is given by

$$Y_{601} = 0.2989 \times R + 0.5866 \times G + 0.1145 \times B. \tag{3.4}$$

The weights for RGB components in Rec. 601 was chosen in 1953 for NTSC RGB phosphors. With the development of HDTV in America and Japan, the old RGB weights are not able to reflect the characteristics of current display devices. Therefore, a new set of RGB weights for computing the Y component has been recommended in Recommendation ITU-R BT.709 (Rec. 709) [81]. The Y component for the new RGB weights in Rec. 709 is then given by

$$Y_{709} = 0.2125 \times R + 0.7154 \times G + 0.0722 \times B. \tag{3.5}$$

Despite Y_{709} being closer to the characteristics of current display devices, Y_{601} is still widely used in image and video compression. Therefore, this thesis adopts Y_{601} for comparison purposes.

The transforms between RGB and the luminance-chrominance color spaces are given in the following equations.

$$\text{RGB} \rightarrow \text{YCbCr} \quad \begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.418 & -0.082 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.6}$$

$$\text{YCbCr} \rightarrow \text{RGB} \quad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.402 \\ 1.000 & -0.346 & -0.715 \\ 1.000 & 1.771 & 0.000 \end{bmatrix} \begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} \tag{3.7}$$

$$\text{RGB} \rightarrow \text{YUV} \quad \begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.8}$$

$$\text{YUV} \rightarrow \text{RGB} \quad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.000 & 1.400 \\ 1.000 & -0.396 & -0.581 \\ 1.000 & 2.029 & 0.000 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \tag{3.9}$$

$$\text{RGB} \rightarrow \text{YIQ} \quad \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.212 & -0.525 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.10}$$

$$\text{YIQ} \rightarrow \text{RGB} \quad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.105 & 1.702 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \tag{3.11}$$

From Equation 3.6 and 3.8 we can note that YCbCr and YUV spaces perform similar color decorrelation as the opponent color process of our visual system: Y is similar to the B-W channel that encodes luminance information; Cb and U are similar to the B-Y channel that encodes blue and yellow color differences; and Cr and V are similar to the R-G channel that encodes the red and green color differences. However, according to Equation 3.10, YIQ does encode color differences, but it does not resemble the opponent color decorrelation of our visual system well.

One thing that should be noted here is that chromatic components range from -127 to $+127$, but for convenience they are normally level shifted to 0 to 255 by adding a shift factor 128.

3.2.3 Digital Image and Video

A digital monochrome image is normally represented by an array of pixels, and the value of each pixel represents the luminance excitation level at that spatial location. Therefore, a digital color image will require three arrays to represent the excitation levels of RGB primaries.

The *Parrots* image shown in Figure 3.5 is a typical RGB color image. Its RGB components are separately shown in Figure 3.6. It is clear to see that all three color components are correlated to each other, as they all contain spatial details of the original *Parrots* image.

Figure 3.5: *Parrots* - a typical RGB color image

Figure 3.6: RGB components of the *Parrot* Image

On the other hand, Figure 3.7 shows the result of transforming the *Parrots* image into YUV color space. We can see only the Y component has the spatial details of the *Parrots* image, while the U and V components contain only the color differences. The histogram comparison in Figure 3.8 also suggests that transforming a color image from RGB to YUV color space can achieve a more compact image data representation. The histograms of RGB components show that RGB pixel values spread over the full range of excitation levels. While, the histograms of YUV components show that the pixel value range of U and V components are relatively concentrated, and only Y component has a full pixel value range.

Figure 3.7: YUV Components of the *Parrot* Image

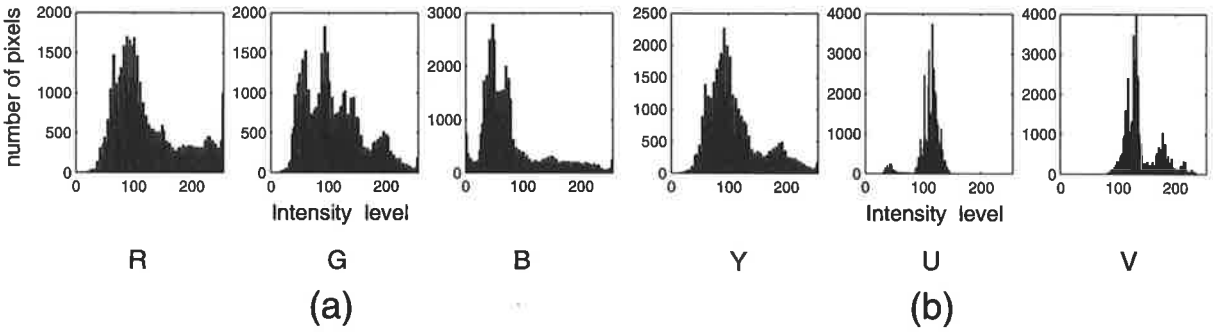


Figure 3.8: Histograms of the *Parrot* image (a) in RGB space (b) in YUV space

3.3 Wavelet Transform

The wavelet transform is a very powerful tool for image compression. This is not only because it is simple to implement, but also because it decomposes visual data in the way that is similar to the multichannel decomposition of our visual system. Therefore, this section will briefly discuss some topics related to the wavelet transform, including the reasons for using the wavelet transform, the mathematical background, practical implementation, its relation to the HVS, and implementation issues.

3.3.1 Why Do We Need the Wavelet Transform?

The reason we need to use the wavelet transform is to decorrelate the visual data, because beside the color correlation discussed in the previous section, an image also has a strong *spatial correlation*. That is, the neighboring pixels in an image are likely to have the same or similar relative intensity values.

For instance, the *Lena* image in Figure 3.9(a) has some smooth regions, such as the back-

ground and the skin, in which pixels tend to have similar values. The correlation can be more clearly visualized from the *cross-correlogram* shown in Figure 3.9(b). This cross-correlogram measures the value of a pixel at location (x, y) on the horizontal axis and the value of the neighboring pixel at $(x, y + 1)$ on the vertical axis. The highly clustered identity line in the correlogram shows that most of the neighboring pixels are of the same or similar intensity values. Knowing the pixel value at a location often will enable us to predict the neighboring pixel values. Therefore, this information redundancy is called *spatial redundancy* and should be removed for efficient data representation. One way of removing spatial redundancy is to represent image data in the form that only stores the differences of pixel values. But this method will depend heavily on the image's entropy, and there is no guarantee that it will achieve good compression results.

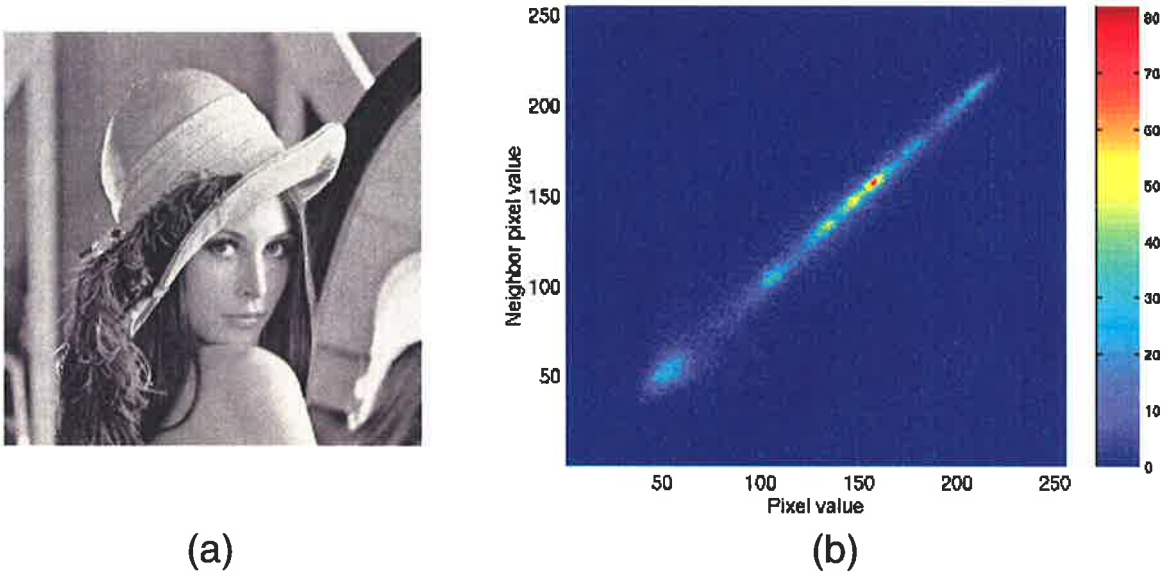


Figure 3.9: (a) 256x256 *Lena* image (b) pixel value correlogram of pixels at location (x, y) and $(x, y + 1)$

A better way for removing spatial redundancy is to transform the image data into a representation that is similar to the visual information representation of our visual system. The wavelet transform is a data transform that is similar to the multichannel models of our visual system [83, 84, 85]. Both the wavelet transform and our visual system decompose visual data into a number of channels that respond to a region of spatial location, a limited band of spatial frequencies, and a limited range of orientations. One of the significant benefits of using wavelet transform is that the transformed image data will have a similar format to the visual data representation in the visual system. Therefore, we not only remove the spatial redundancies of image data, but also can take advantage of the visual insensitivities discussed in the previous chapter

to remove more visually unimportant information for better compression results. More of the similarities of the wavelet transform to the human visual system will be discussed later in this section, but before that let's take a brief look at the mathematical background.

3.3.2 Mathematical Background

Although the Fourier transform is a very useful tool for signal analysis, it is only suitable for analyzing a signal's frequency-domain behaviors, due to its poor time-localization basis functions, *sine* and *cosine*. The time-bandwidth product is bound by the *Heisenberg inequality* [86], as indicated by

$$\text{Time-Bandwidth product} = \Delta t \Delta f \geq \frac{1}{4\pi}. \tag{3.12}$$

This inequality means that the tradeoff can only be made between time resolution and frequency resolution. Therefore, the Fourier transform is only suitable for analyzing stationary signals.

On the other hand, the basis functions of the wavelet transform are chosen to have a compact support and a variable support width. These properties allow the wavelet transform to be able to analyze both the frequency- and time-domain behaviors of a signal. Therefore, the wavelet transform is suitable for analyzing both stationary and non-stationary signals [87, 88].

The principle of the wavelet transform is to represent a signal $f(x)$ by the superposition of a wavelet set that is obtained from the *mother wavelet*. The equation for a set of wavelets is given by

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right), \tag{3.13}$$

where ψ is the mother wavelet, a is the scaling coefficient, and b is the translation coefficient. Coefficients a and b are chosen such that $a > 0$ and $b \in R$ [89]. The factor $\frac{1}{\sqrt{a}}$ in Equation (3.13) ensures that the wavelet set has a uniform norm [90]. The mother wavelet ψ needs to have a compact support, that is, ψ has to die away in few oscillations, and satisfy $\int \psi(t)dt = 0$. This will ensure that the mother wavelet has good time-localization.

The following will give a brief summary of continuous wavelet transform, wavelet series expansion, and multiresolution analysis. For more details, readers are referred to [90, 91].

Continuous Wavelet Transform

The continuous wavelet transform is normally applied for analyzing natural continuous signals. Consider a continuous signal $f \in L_2(R)$, then the wavelet transform of f is then given by

$$W_f(a,b) = \langle f(t), \psi_{a,b}(t) \rangle = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(t)\psi_{a,b}(t)dt. \tag{3.14}$$

This equation can be viewed as the calculation of linear projections to the basis, $\psi_{a,b}$, that spans the space $L_2(R)$. The linear projections are the transform coefficients, $W_f(a,b)$, which represent the weights of contribution from the basis, $\psi_{a,b}$, to the original function, $f(x)$.

For the inverse transform to exist, the choice of ψ has to satisfy

$$C_\psi = \int_{-\infty}^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega < +\infty. \tag{3.15}$$

This is called the *admissibility criterion* [87, 89, 92]. It means that $\psi(t)$ decays faster than $|t|^{-1}$ for $t \to \infty$. If this condition is satisfied, then the inverse wavelet transform is given by

$$f(t) = \frac{1}{C_\psi} \int_0^{\infty} \frac{da}{a^2} \int_{-\infty}^{\infty} W_f(a,b)\psi_{a,b}(t) db. \tag{3.16}$$

Wavelet Series Expansion

In practical applications, factors a and b in Equation 3.14 are chosen to be discrete, rather than continuous. Therefore, if we set $a = {a_0}^m$, $b = n{b_0}{a_0}^m$, $(m,n \in Z)$, $a_0 > 1$, and $b_0 > 0$, then the discretized set of wavelets becomes

$$\psi_{m,n}(t) = a^{-\frac{m}{2}} \psi(a_0^{-m} t - nb_0). \tag{3.17}$$

The forward *wavelet series expansion* of $f(t)$ is given by

$$C_{m,n}(f) = \langle \psi_{m,n}, f(t) \rangle = \int_{-\infty}^{\infty} \psi_{m,n}(t) f(t) dt, \tag{3.18}$$

and $f(t)$ can be reconstructed by

$$f(t) = \sum_{m,n} C_{m,n}(f) \psi_{m,n}(t). \tag{3.19}$$

If we choose $a_0 = 2$ and $b_0 = 1$, then the set of ψ_{mn} forms an orthonormal basis in $L_2(R)$.

Multiresolution Analysis

For multiresolution analysis, we need two functions: one is the wavelet function ψ, discussed previously; and the other is the scaling function, ϕ. The scaled and translated versions of ψ and ϕ, with $a = 2, b = 1$, and $n \in Z$, are given by

$$\Psi_{m,n}(t) = \frac{1}{\sqrt{2^m}} \psi(2^{-m} t - n), \tag{3.20}$$

$$\Phi_{m,n}(t) = \frac{1}{\sqrt{2^m}} \phi(2^{-m} t - n). \tag{3.21}$$

For a fixed m, the family of $\phi_{m,n}$ forms an orthonormal basis, and $\phi_{m,n}$ spans the space V_m. The space V_m represents a sequence of successive approximated subspaces,

$$\cdots V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \cdots, \tag{3.22}$$

and each subspace has a resolution 2^m. On the other hand, for a fixed m, $\psi_{m,n}$ will span space W_m, which is the orthogonal complement of V_m in V_{m-1}, ie,

$$V_m \oplus W_m = V_{m-1}. \tag{3.23}$$

Therefore, the coefficients of $\langle \psi_{m,n}, f \rangle$ represent the information required when going from an approximation of f at resolution 2^m to a finer approximation of f at resolution 2^{m-1} [89].

In practice, the function f is sampled discretely, and the sampled coefficients $a_{0,n}$ are the approximate coefficients at the highest resolution. The space associated with $a_{0,n}$ is V_0. The coefficients of projections onto successive coarser spaces V_1 and W_1 are respectively given by

$$a_{1,n}(f) = \sum_k h_{2n-k} a_{0,k}(f), \tag{3.24}$$

$$c_{1,n}(f) = \sum_k g_{2n-k} a_{0,k}(f), \tag{3.25}$$

where $g_n = (-1)^n h_{-n+1}$ and $h_n = 2^{1/2} \int \phi(x-n)\phi(2x)dx$. The general equations for going from space V_{m-1} to spaces V_m and W_m then becomes

$$a_{m,n}(f) = \sum_k h_{2n-k} a_{m-1,k}(f), \tag{3.26}$$

$$c_{m,n}(f) = \sum_k g_{2n-k} a_{m-1,k}(f). \tag{3.27}$$

On the other hand, the general equation for constructing finer scale coefficients from space V_m to space V_{m-1} is given by

$$a_{m-1,l}(f) = \sum_n [h_{2n-l} a_{m,n}(f) + g_{2n-l} c_{m,n}(f)]. \tag{3.28}$$

In practice, multiresolution analysis is implemented by a pair of orthogonal FIR (finite impulse response) filters, ie. h as the low-pass filter and g as the high-pass filter. However, in order to have a linear phase for easy cascading, the wavelet basis is sometimes chosen to be biorthogonal instead of orthogonal. The reconstruction equation is then given by

$$a_{m-1,l}(f) = \sum_n [\tilde{h}_{2n-l} a_{m,n}(f) + \tilde{g}_{2n-l} c_{m,n}(f)], \tag{3.29}$$

where $\tilde{h}$ and $\tilde{g}$ are chosen to have a perfect reconstruction, and satisfy

$$\tilde{g}_n = (-1)^n h_{-n+1}, \tag{3.30}$$

$$g_n = (-1)^n \tilde{h}_{-n+1}, \tag{3.31}$$

$$\textstyle\sum_n h_n \tilde{h}_{n+2k} = \delta_{k,0}. \tag{3.32}$$

3.3.3 Practical Implementation

The practical implementation of the wavelet transform as a FIR filter bank is under the assumption that a 2D wavelet transform can be separated into two 1D wavelet transforms.

An example of such a filter bank implementation for the forward wavelet transform (FWT) is shown in Figure 3.10. Image $f(x, y)$ is first filtered and down-sampled by 2 in the horizontal direction, ie. along *x-axis*. The results are two subbands with half of the original size: the low-pass filtered subband contains low spatial frequency information, and the high-pass filtered subband contains high spatial frequency details in the *vertical* direction. After that, these two horizontally filtered subbands are further vertically filtered along *y-axis*, and down-sampled. The results are four subbands of a quarter of the original size: *LL* subband contains low spatial frequency information, *LH* subband contains *horizontal* high frequency details, *HL* subband contains *vertical* high frequency details, and *HH* subband contains *diagonal* high frequency details. The labeling notation is that the first letter indicates the low-pass (*L*) or high-pass (*H*) filtering in horizontal direction, and the second letter indicates the low-pass or high-pass filtering in the vertical direction.

If these transform procedures are repeatedly applied to the *LL* subband, the result is *multiresolution analysis*. The down-sampling factor 2 ensures that no data expansion occurs and the transform is a *non-expensive* transform. Down-sampling by 2 is often called *critical sampling*. The filter bank implementation of the inverse wavelet transform (IWT) is shown in Figure 3.11. The inverse wavelet transform is in the reverse order of the forward transform.

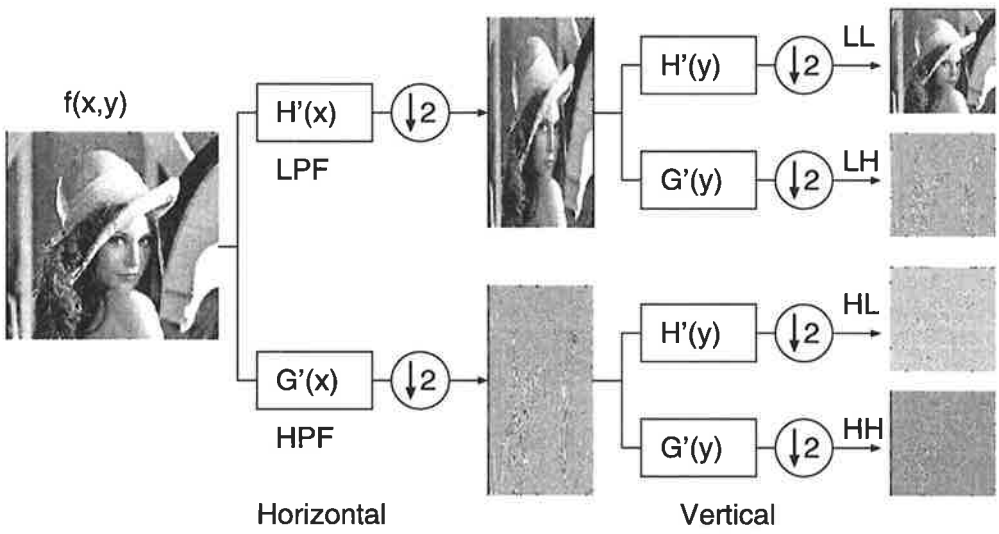


Figure 3.10: FIR Filter Implementation of the forward wavelet transform (FWT)

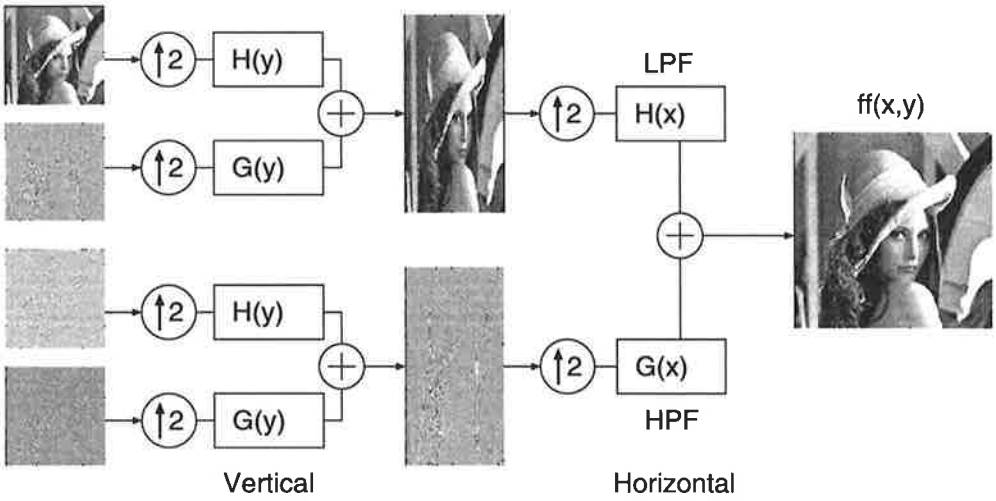


Figure 3.11: FIR Filter Implementation of the inverse wavelet transform (IWT)

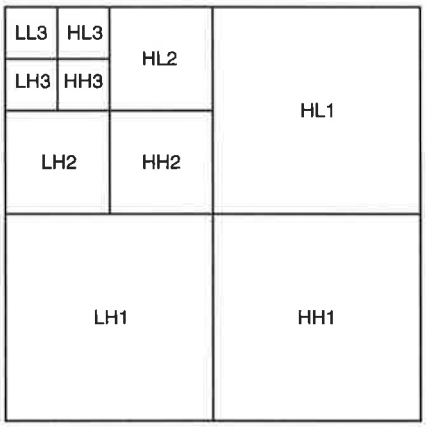


Figure 3.12: Subband spatial orientation after 3-level wavelet transform

Figure 3.12 shows an example of the subband spatial orientation we will see on a 3-level wavelet transformed image. In addition to the subband labels, the number after each label indicates the wavelet transform level, like 1 for the first level and 2 for the second level. Figure 3.13 shows the results of transforming the *Sydney* image to different transform levels. From the figure we can see that the high frequency subbands exhibit very strong directional selectivity behavior.

Figure 3.14(a) shows the *Lena* image after the application of the 2-level wavelet transform. It shows that the *LL* subband contains most of the image information, while the high frequency subbands only contain the spatial differences of the original image. This information packing property of the wavelet transform can also be seen more clearly from the histogram comparison of the *LL* subband and high frequency subbands. The histogram of the *LL* subband in Figure 3.14(b) shows that the *LL* subband coefficients have large magnitudes, as the magnitude distri-

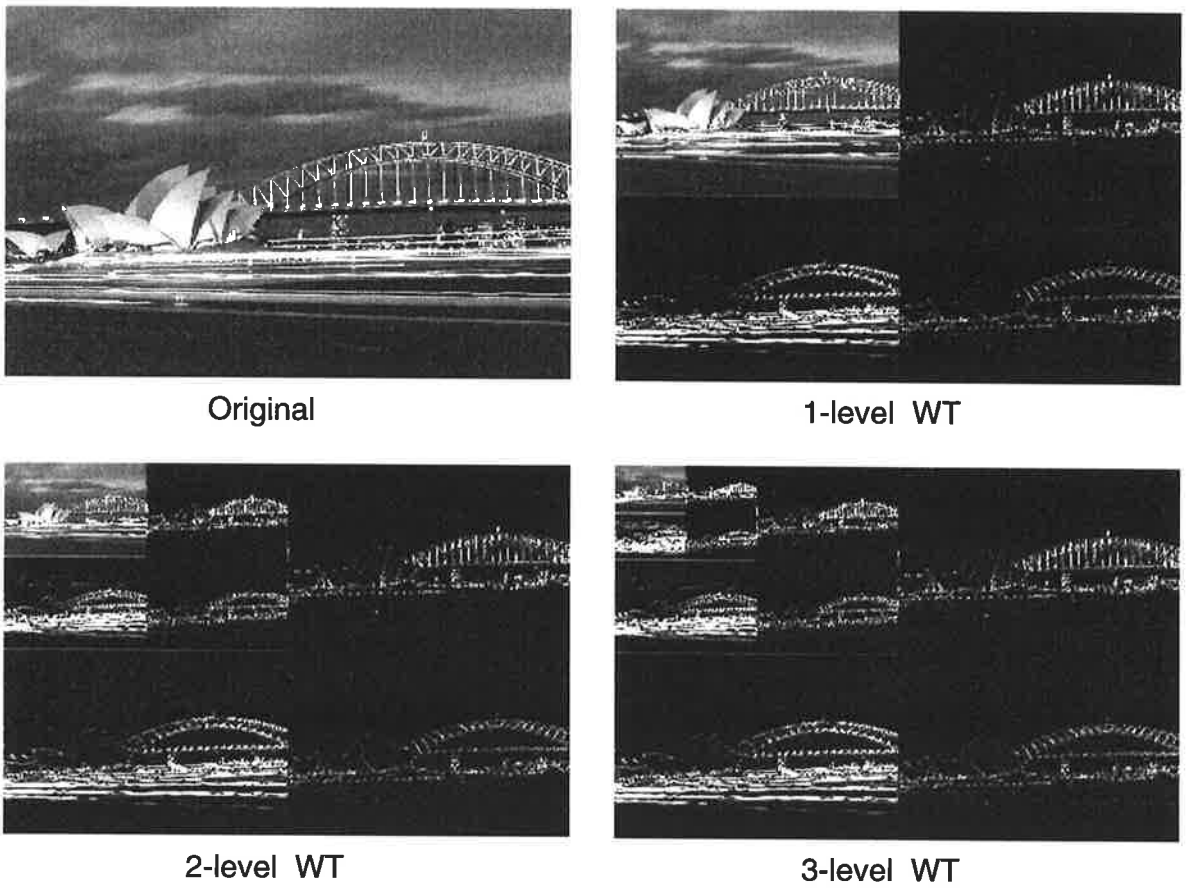


Figure 3.13: The *Sydney* image at different wavelet transform levels

bution scatter around all range of histogram. On the other hand, the histogram of high frequency subbands in Figure 3.14(c) shows that most of the high frequency coefficients are very small, as the magnitude distribution concentrates around zero.

3.3.4 Wavelet Transform and the HVS

As mentioned in Chapter 2, the receptive fields of the visual cortex neurons are tuned to respond to different orientations and frequency ranges. The neurons also have varied sensitivities to the visual data. Therefore, for achieving a better compression, we should decompose the visual data into a multichannel representation similar to that in the human visual system, and remove the information that is less important to the visual system.

One way to decompose the visual data is to use the cortex transforms discussed previously. Although cortex transforms resemble the multichannel decomposition of the human visual system exactly, they are very computationally expensive and too complicated for implementation. Also, cortex transforms adversely increase the amount of transform data [51], so they are not

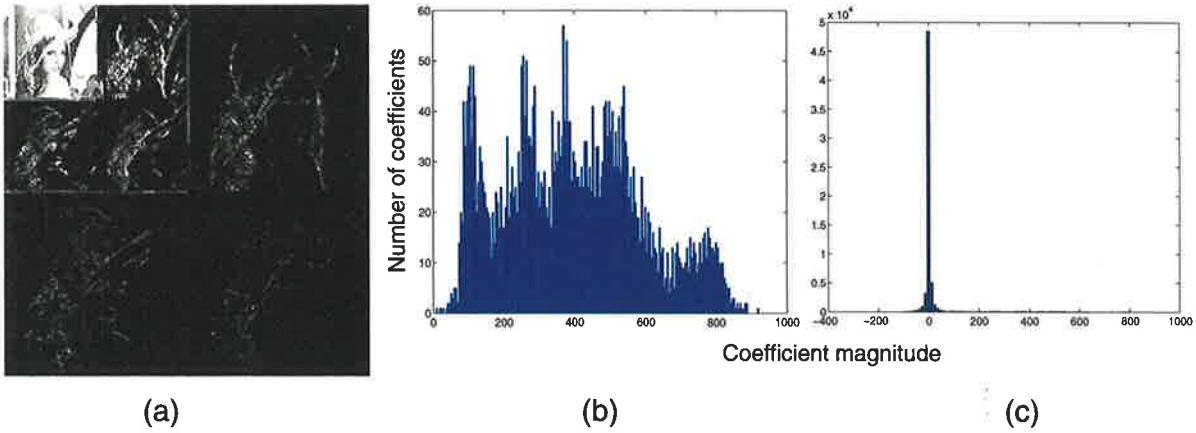


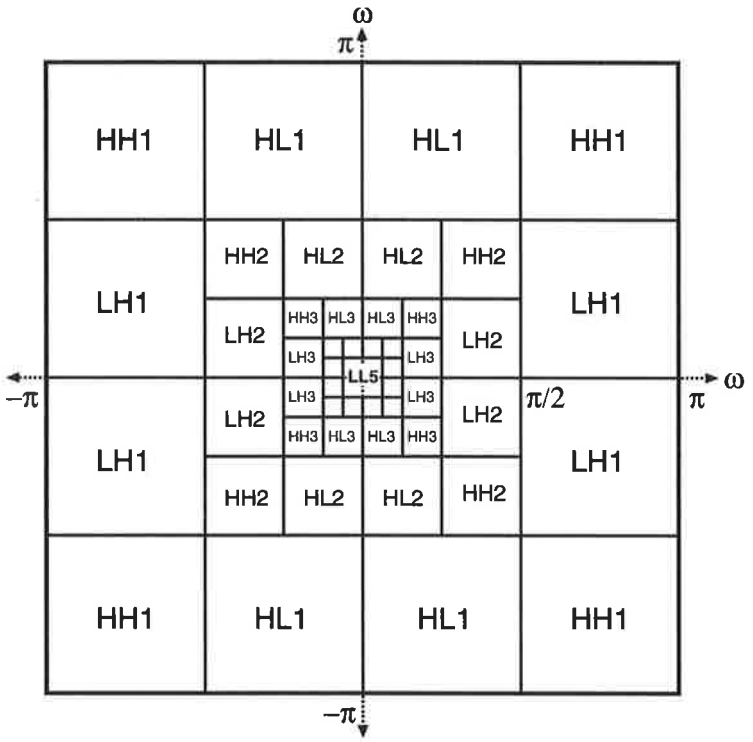
Figure 3.14: (a) The *Lena* image after 2-level WT (b) histogram of the *LL* subband (c) histogram of all high frequency subbands

suitable for image and video compression.

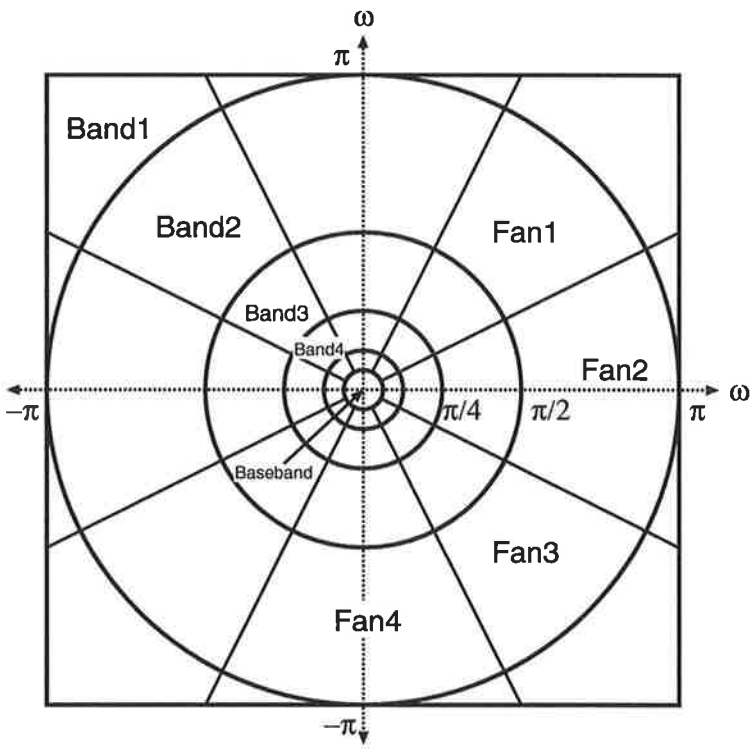
A better way to decompose visual data into multichannel representation without increasing the amount of transformed data is to use the wavelet transform. The wavelet transform not only decomposes visual data into the format that is closely related to the cortex decomposition of the human visual system, but also maintains the same data amount as the original image, due to critical down-sampling. However, the most important feature is that the wavelet transform is easy to implement and computationally efficient.

Figure 3.15(a) shows the wavelet subband orientation for a 5-level decomposition in the frequency domain. The wavelet decomposition is similar to the typical cortex transform of the human visual system shown in Figure 3.15(b). Both transforms have 1 octave of frequency bandwidth for each frequency resolution, so they have frequency channels of similar decomposition resolution. For example, the wavelet subbands in transform level 1 are analogous to the combination of *Band1* and *Band2* in the cortex transform, and the *LL5* subband is analogous to the *Baseband* of the cortex transform.

Furthermore, the wavelet transform has similar orientation selective channels to the cortex transform. The wavelet transform has three types of subbands that correspond to the three different orientation selective channels, ie. the *LH*-, *HL*-, and *HH*-subbands. Like *Fan2* in the cortex transform, the *LH*-subband has horizontal selectivity. Similarly, the *HL*-subband has vertical selectivity like *Fan4* in the cortex transform. However, the wavelet transform has only one diagonal-selective channel, the *HH*-subband, which effectively combines the responses of *Fan1* and *Fan3*. Nevertheless, the limited number of orientation selective channels for the wavelet transform is the cost we are willing to pay for using a more efficient transform.



(a)



(b)

Figure 3.15: (a) Wavelet transform (b) a typical cortex transform model of the HVS

3.3.5 Implementation Considerations

When applying the wavelet transform to visual data compression, there are two implementation considerations which directly affect the compression quality and which must be examined closely.

Signal Extension Methods

The first consideration is the selection of signal extension methods. Since images and video frames are finite-length signals, there is a need to extend the coefficients at the boundaries for the filter bank implementation. There are three possible extension methods which can be used for the implementation.

The first method is *zero-padding* which pads zeros at the boundaries as in the diagram shown in figure 3.16(b). Zero-padding is easy to implement but gives poor coding results. This is because the abrupt transition at the boundaries adds variance to the high-pass subbands and reduces the coding gain of the filter bank.

The second method is *periodic extension* as in the diagram shown in Figure 3.16(c). Periodic extension is similar to the extension for FFT (Fast Fourier Transform) circular convolution. Periodic extension was first suggested by Woods and O'Neil [93] and later by Smith and Eddins [94]. This is a reasonable assumption for finite length signals but discontinuities still occur at the image boundaries and variance still happens to the high-pass subbands.

The third method is *symmetric extension*. That is, coefficients at the boundaries are symmetrically extended as in the diagram shown in Figure 3.16(d). Symmetric extension was first introduced by Smith and Eddins [94]. Because of a smoother transition at the image boundaries, symmetric extension gives better compression results than the other two methods [91](p.341).

However, symmetric extension does not always work. This is due to different combinations of symmetric extension methods and filters of different symmetries and phases. Only certain combinations will be able to maintain non-expensive wavelet transform and produce symmetric subbands. Martucci et al. [95, 96], Kiya et al. [97], and Bamberger et al. [98] presented several methods to select the right symmetric signal extensions for the filters to generate non-expansive symmetric subbands. They also extended their work from two filter bank applications to multirate filter bank applications. However, the complete classification for the right combination of symmetric extension methods and filters was presented by Brislawn [99, 100].

Figure 3.17 shows images which were compressed [1] to 96:1 ratio or 0.25 bit per pixel (bpp)

[1] Images were compressed by the LZC algorithm which will be discussed in detail in Chapter 6.

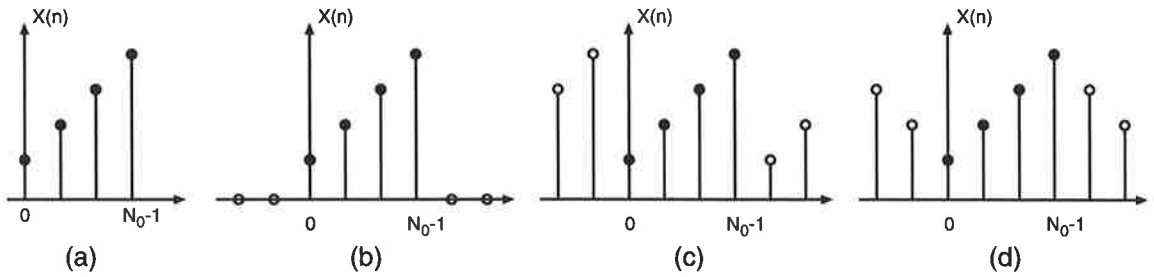


Figure 3.16: Signal extension method (a) Original finite-length signal (b) Zero-Padding extension (c) Periodic Extension (d) Symmetric Extension

using different extension methods. The results show that the zero-padding method produces large errors at the boundaries, as shown in Figure 3.17(b), so it is not appropriate for image or video compression. The periodic extension method does not yield any errors at the boundaries when the image is not quantized, due to the *perfection reconstruction* property of the filter bank. However, errors do occur at the boundaries of the compressed image because the periodic extension method still contributes discontinuities at the boundaries, as shown in Figure 3.17(c). The symmetric extension method gives the best subjective (perceptual) quality because of the smoother boundaries, as shown in Figure 3.17(c). The mean square error (MSE) and the peak signal to noise ratio (PSNR) of these three images are tabulated in table 3.2. This table shows that symmetric extension method also gives the best objective (numerical) results. The methods of calculating these results will be discuss in more detail in Section 4.3.2.

	Zero-Padding	Periodic Extension	Symmetric Extension
MSE	370.77	160.72	141.82
PSNR	22.44	26.07	26.61

Table 3.2: MSE and PSNR (dB) of 96:1 WT compressed images using different extension methods

Wavelet Filters

The second implementation consideration is the selection of wavelet filters, and the selection is based on three criteria, namely *linear phase*, *regularity* and *filter size*.

Linear phase filters are desirable for image compression because they can be easily cascaded in pyramidal filter structures without the need for phase compensation [89] and enable the use of the preferable symmetric extension at the signal boundaries [101]. Filters have linear phase only when their coefficients are symmetric or antisymmetric around the center coefficient [91](p.10).

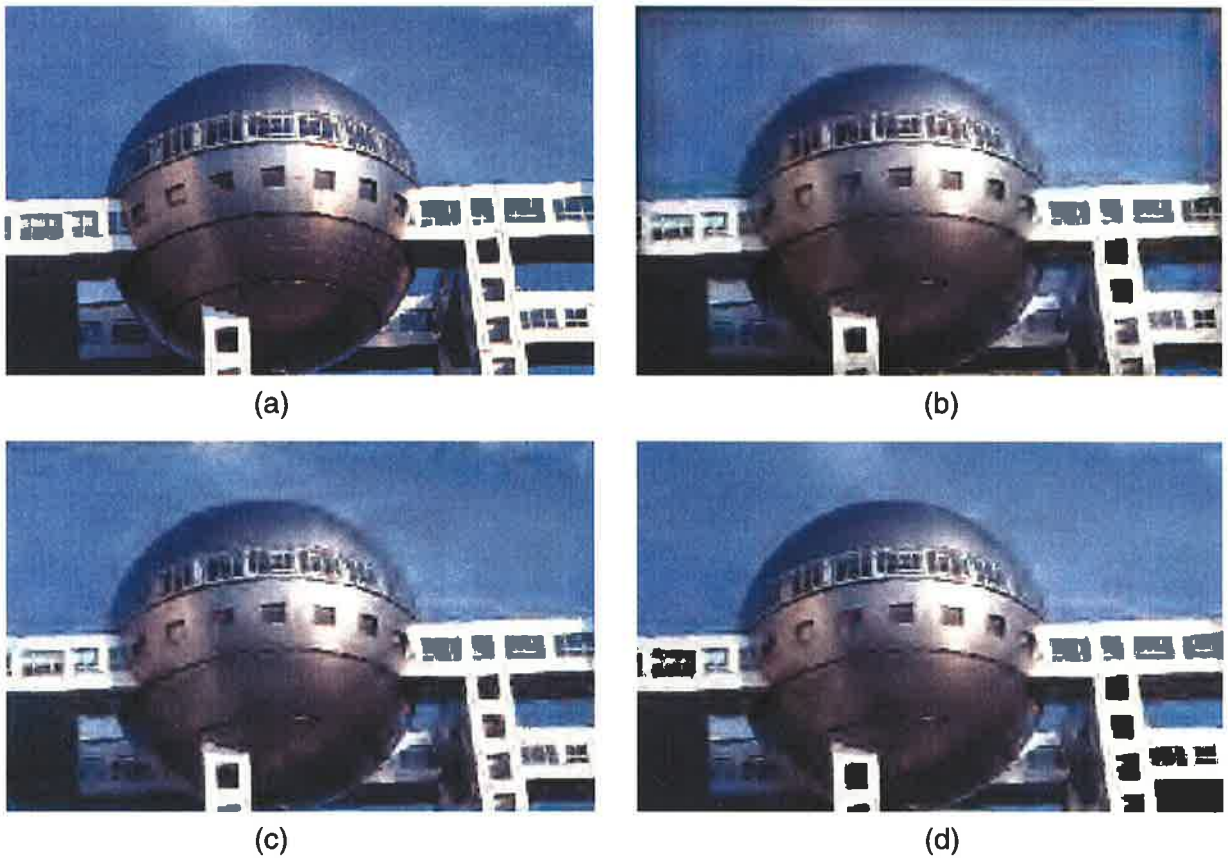


Figure 3.17: 96:1 WT compressed images using different extension methods (a) Original (b) Zero-Padding (c) Periodic (d) Symmetric

Therefore, linear phase and orthogonality are mutually exclusive, as the coefficients of orthogonal filters are neither symmetric nor antisymmetric. Only biorthogonal filters have linear phase by relaxing the orthogonality and use biorthogonal bases having either symmetric or antisymmetric filter coefficients. As a result, biorthogonal filters are more suitable than orthogonal filters for image compression.

The regularity of filters is crucial for image compression because images are mostly smooth, and regular filters are more desirable in analyzing images [89, 102]. A filter is said to be *regular* if it has a certain number of *zeros* at the aliasing frequency π (or at -1 in z-domain) and its iterated function converges to a smooth continuous function [88, 87]. The more zeros the filter has the smoother the iterated continuous function and, therefore, the more regular the filter. The number of zeros that a filter has is closely related to its length. A longer filter normally has more zeros at π [103]. However, the length of filters also must be limited under the constraint that they must not increase computational complexity and error spreading too much . The increase of computational complexity is due to the increase numbers of filter coefficients, and error spreading is due to the filter spreading quantization errors across neighboring coefficients

in the horizontal and vertical range of $-N/2$ to $N/2$, where N is the filter length. If images are quantized coarsely, the error spreading can cause *ringing artifacts* at high coefficient transitions, such as object contours or boundaries [91](p.372).

The discussion on the selection of a biorthogonal filter bank system can be found in [103] and [91](p.372), and the key points are summarized as follows. First, a regular filter produces less noise, so the synthesis low-pass filter should be regular. An irregular synthesis low-pass filter will cause blocky artifacts on the reconstructed image. Second, for a two-channel biorthogonal filter bank, the synthesis filters should have a longer low-pass filter for a smoother result and a shorter high-pass filter for minimizing the ringing artifacts. Third, the size of analysis filters is not so critical, but the low-pass analysis filters should have at least one zero at π to stop high frequency leakage, and the analysis high-pass filter should have at least one zero at 0 to stop low frequency leakage. High frequency leakage will reduce the coding gain, and low frequency leakage will cause the low frequency information to be coarsely quantized.

Orthogonal filters are simple to implement but they do not have linear phase, so the reconstructed images and video will have more errors. On the other hand, biorthogonal filters have a linear phase and are symmetric; therefore, they are more favorable for image and video compression.

3.4 Entropy Coding

After image and video compression, the coded bit-stream can be grouped into an ensemble of symbols with some random probability redundancies. Those probability redundancies can be measured by Shannon's famous equation [1]

$$H = -\sum_{i=1}^{n} p_i log_2 p_i, \tag{3.33}$$

where p_i is the probability of the ith symbol. The quantity H in the equation is called the entropy of a set of probabilities or the measurement of the uncertainty of an ensemble of symbols. The unit of H is $bits$, as the base of the logarithm is 2. The entropy determines the lowest bound of the average bits per symbol without losing any information of the ensemble. Intuitively, the coded bit-stream can be further losslessly compressed so that the average number of bits per symbols is as close to the entropy as possible. There are two widely used entropy coding methods for removing probability redundancies, namely *Huffman Coding* and *Arithmetic Coding*.

3.4.1 Huffman Coding

Based on Shannon's theory, Huffman [104] developed an efficient entropy coding method which is now known as Huffman Coding for coding a finite number of symbols to approach their entropy. However, there are two disadvantages of Huffman coding.

The first disadvantage is that Huffman Coding does not adapt to the change of symbol probabilities. Therefore, the symbol probabilities need to be known *a priori* by the encoder and the decoder. One solution is to have a pre-coding scan which will run through the entire symbol set to obtain to the symbol probabilities for the encoder. The symbol probabilities can then be transmitted to the decoder by embedding in the coded bit-stream. However, performing a pre-coding scan is not always feasible, especially in real-time applications, and embedding symbol probabilities in the coded bit-stream increases coding overheads. A better solution is to use a pre-trained probability model for both the encoder and the decoder. Nevertheless, using a pre-trained model is likely to have a sub-optimal entropy coding, due to inexact probability modeling.

The second disadvantage is that Huffman coding can only assign an integer number of bits to the coding symbols. In the case when the symbol probabilities are the reciprocal of a power of 2, the integer number assignment to the symbol is rate-efficient. However, if symbol probabilities are real numbers, then the integer number assignment becomes inefficient.

3.4.2 Arithmetic Coding

The two defects of Huffman Coding were solved by Arithmetic Coding, first introduced by Rissanen [105] and Pasco [106], and later by Witten et al. [107]. Arithmetic coding separates the probability model from the coding procedure so the model can be adaptively updated to different sources. In addition, Arithmetic Coding also allows the symbols to be coded by a real number of bits.

The basic idea of Arithmetic coding is that symbols are assigned to the subintervals between the unit interval $[0, 1)$. Each subinterval is specified by the start-point α and the end-point β, ie. $[\alpha, \beta)$, and each subinterval length, l, is proportional to the symbol probability.

Here is a simple encoding example. Table 3.3 lists four source symbols, $(x, y, z, !)$, and their corresponding probabilities. Symbol ! is the EOF (end of file) symbol, which marks the end of coding. Suppose the symbol stream being coded is $yyxz!$, and the corresponding coding procedures are shown in Figure 3.18.

First, the unit interval $[0, 1)$ in divided into interval $[0, 0.1)$, $[0.1, 0.3)$, $[0.3, 0.6)$ and $[0.6, 1)$ for symbol !, z, y and x, respectively. The first symbol received by the encoder is y, so the new coding interval is narrowed down to $[0.3, 0.6)$. This new interval is then sub-divided into four sub-intervals according to symbol probabilities. The second symbol received is y, so the new coding interval is narrowed down to $[0.39, 0.48)$. This new interval is again sub-divided. The coding interval is narrowed and sub-divided repeatedly until the encoder receives symbol !. The calculation of coding intervals for this example is tabulated in Table 3.4. The last coding interval after receiving symbol ! is $[0.4476, 0.44832)$. Therefore, the symbol stream $yyxz!$ can be represented by a number r which satisfies $0.4476 \leq r \in R < 0.44832$.

Symbols	x	y	z	!
Probability	0.4	0.3	0.2	0.1

Table 3.3: Source symbols and their probabilities

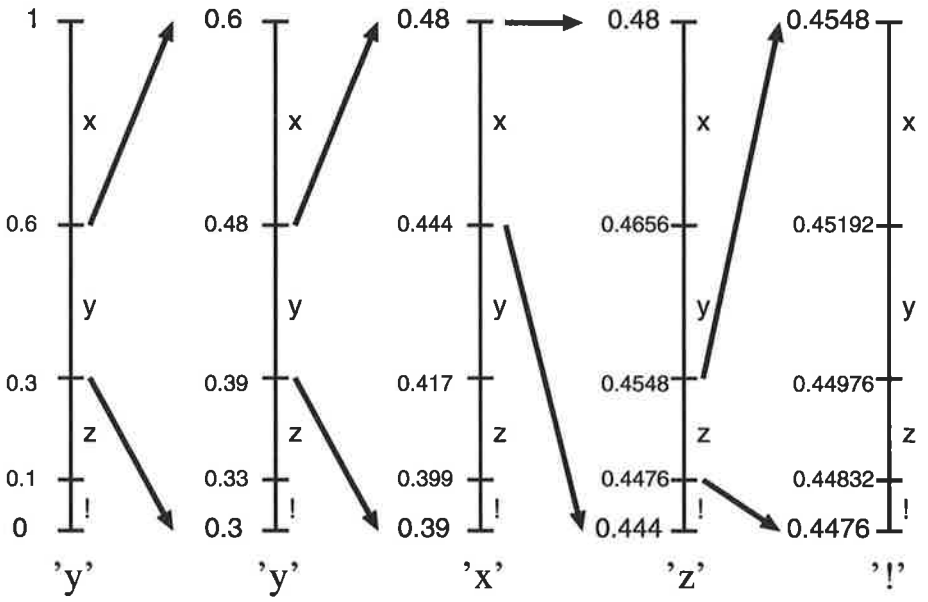


Figure 3.18: Arithmetic Coding procedures for symbol stream $yyxz!$

Different Arithmetic Coding methods have different ways of finding the number r. Here the number r was found by the method shown in [108](p.123). Firstly, find an integer t which satisfies

$$\frac{1}{2^t} \leq l = \text{lengh of interval}, \tag{3.34}$$

so let $t = 11$ for the interval of 0.00072 in this case. Secondly, find an integer c which satisfies

$$0.4476 \leq \frac{c}{2^t} = \frac{c}{2048} < 0.44832 \text{ or } 916.6848 \leq c < 918.15936, \tag{3.35}$$

next symbol	start-point α	end-point β	length of interval, l
	0	1	1
y	0.3	0.6	0.3
y	0.3+0.3×0.3=0.39	0.3+0.6×0.3=0.48	0.09
x	0.39+0.09×0.6=0.444	0.39+0.09×1=0.480	0.036
z	0.444+0.036×0.1=0.4476	0.444+0.036×0.3=0.4548	0.0072
!	0.4476+0.0072×0=0.4476	0.4476+0.0072×0.1=0.44832	0.00072

Table 3.4: Coding intervals for stream $yyxz!$

so let $c = 918$. Therefore, the real number r is

$$r = \frac{918}{2048} = 0.448242178 = 0.0111001011_2. \tag{3.36}$$

Hence, the Arithmetic coded bit stream for $yyxz!$ is 0111001011.

The Arithmetic decoding procedures are in the exact same order as encoding. Assuming the decoder knows the source symbol probability *a priori*, so the decoder can first divide the unit interval into the corresponding four sub-intervals. The decoder first receives the encoded bit-stream 0111001011, which represents the real number 0.448242178. Since this number lies in the range of $'y'$, the decoder instantly knows the first symbol is $'y'$. After that, the coding interval is narrowed down to $[0.3, 0.6)$ and sub-divided into four sub-intervals. The encoder compares the real number with the four new sub-intervals, and finds out that the real number lies in the y's interval, so the decoder outputs the second symbol as $'y'$. The decoder iteratively narrows and sub-divides the interval, compares the real number with the sub-intervals, and outputs the corresponded symbols until the EOF is found.

Although the above example is based on the fixed probability model, in practice the probability model can be adaptive and vary during the coding depending on the input source. In addition, due to the limitation of the computing resources, the precision of dividing the word length of a real number is limited, and there are some chances that underflow and overflow may occur. The underflow and overflow situations can be resolved by imposing some constraints on the word length of the coded real number, and by rescaling of the frequency counts of the probability model. More details of such Arithmetic Coding implementation issues and treatments are discussed in [107].

Since the probability model of visual data is a random process, it is not possible to find a probability model to represent all the images and video sequences. An adaptive probability model of an entropy coder will be more appropriate than a fixed model. Therefore, Arithmetic Coding is more suitable than Huffman Coding for reducing the entropy of the compressed image or video bit-stream.

Chapter 4

Compression Schemes and Quality Assessment

4.1 Introduction

This chapter will briefly review the compression schemes that are currently used for visual data compression. Those compression schemes normally exploit either the visual insensitivities or visual data redundancies, or both, which have been discussed in the preceding chapters. This chapter will also review some methods used to assess the quality of the compressed visual data. The quality assessment can be used to predict the subjective quality of the image viewed and to improve the compression schemes.

4.2 Visual Compression Schemes - Review

Currently all visual compression techniques can be broadly grouped into four categories: sample-based methods, block-based methods, fractal methods, and multiresolution methods.

4.2.1 Sample-Based Methods

For a sample-based compression method, the image samples are compressed on a sample-by-sample basis. A sample-based compression method is a scalar predictive coding which predicts image samples either in the spatial domain or in the frequency domain. It is based on closely correlated neighboring pixel values — spatial redundancy. The differential pulse code modu-

lation (DPCM) method shown in Figure 4.1 is a simple example. The predicted pixel value is given by

$$p_{i,j} = \sum_{(k,l)\in N} w_{k,l} q_{i-k,j-l}, \tag{4.1}$$

where N is a group of causal neighboring pixels, $w_{k,l}$ is the prediction weight of the pixel at position (k,l), and $q_{i,j}$ are the quantized coefficients from which to predict. The values of the prediction weights, $w_{k,l}$, can be either fixed or adaptively estimated. The prediction error $e_{i,j}$ is then given by

$$e_{i,j} = x_{i,j} - p_{i,j} \tag{4.2}$$

where $x_{i,j}$ is the original pixel at coordinates (i,j).. Since neighboring pixels are spatially correlated, the prediction errors, $e_{i,j}$, generally will be small, and have a lower entropy than the original signal x. As a result, the quantized values, $q_{i,j}$, will require fewer coding bits. However, this sample-based method may not always work. If there are too many high spatial frequency activities in an image, this sample-based method may fail to predict samples, and would have an inefficient coding result.

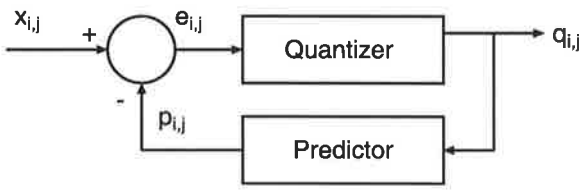


Figure 4.1: DPCM encoding model

4.2.2 Block-Based Methods

Block-based coding methods can be further separated into two classes: spatial-domain block coding and transform-domain block coding. In spatial-domain block coding, the pixels are grouped and compressed in a block-by-block fashion. An example of spatial-block-based coding is vector quantization (VQ) [109]. A VQ encoder takes a block of data and assigns codewords to each block. Since the codewords are also available to the decoder, only the indices of codewords are sent. The result is a very sparse representation of the original data. However, a VQ coder normally has a long coding time, due to the exhaustive search for matching codewords to the coding blocks. In contrast, the VQ decoding speed is comparatively faster than the encoding. Therefore, the VQ method is only suitable for those applications in which fast coding is not important.

On the other hand, in transform-domain block coding, three main coding procedures are performed. Firstly, the pixels are grouped into blocks, and the blocks are transformed to the

frequency domain. Since the transform decorrelates and compactifies block pixels, many of the transform coefficients will be close to zero. Secondly, a quantizer identifies and allocates more coding bit-budget to the coefficients which are visually more important to the viewers. Finally, the quantized coefficients are entropy encoded according to their occurrence probabilities. The coding process can be expressed as

$$\hat{X} = T^{-1}Q(TX), \tag{4.3}$$

where T is the transformation matrix, Q is the quantizer, T^{-1} is the inverse matrix of T, and $\hat{X}$ is the decoded version of the input image X. The optimal transform in the sense of achieving greatest decorrelation is the Karhunen-Loève transform (KLT) [110](p.535), but it is too computationally inefficient for practical applications and also has image dependent basis functions. A better option is the discrete cosine transform (DCT) [111]. DCT is a close approximation of KLT and is widely used in block-based coding. The basis functions of DCT are image independent and the transform of DCT is highly regular. Both the JPEG image compression standard [112, 113, 114] and the MPEG video compression standard [115, 116] use DCT as their transform basis. However, the DCT-based methods are only suitable for medium to high bit-rate applications. At low bit-rate the decoded images and video will have unpleasant blocky artifacts.

4.2.3 Fractal Methods

Fractal image compression methods, introduced by Barnsley [117, 118] and Jacquin [119, 120], are based on the theory of iterated function systems (IFS) [121]. Later Levy and Wilson showed the relation of fractal coding to wavelet coding and used wavelet VQ for both 2D image and 3D video compression [122, 123]. The fundamental idea of fractal image compression is that the image data naturally contains blocks of similar details at the same or different scales, and at the same or different orientations. That is, based on this *self-transformability* the image can then be represented by a group of source blocks extracted from the image itself, together with the location and the transformation indices of the source blocks. However, the fractal compression methods are very computationally expensive and are not suitable for either real-time applications or hardware implementations.

4.2.4 Multiresolution Methods

Multiresolution methods are based on the exploitation of the visual insensitivities of frequency decomposed images. The decomposition is done by a unitary transform which preserves the

input bit-rate. Different to the block transform, such as DCT, the multiresolution decomposition divides the whole image into multiple frequency subbands using a combination of low-pass and high-pass filters. Therefore, this type of compression is also known as *Subband Coding*, which was first applied to image compression by Woods and O'Neil [93] and later by Gharavi and Tabatabai [124]. The relationship between subband filtering and wavelet decomposition was explored by Mallat [125]. Since then the wavelet transform has been extensively applied to multiresolution image and video compression. At low bit-rates, the wavelet compression scheme usually can achieve much better coding quality than the standard DCT schemes. Among all of the proposed wavelet compression schemes, the *zerotree* scheme is the most efficient in terms of coding complexity and coding results. Currently, zerotree is the benchmark of image compression algorithms. Zerotree theory was first introduced by Lewis and Knowles [126], and later by Shapiro [127, 128]. Both the image and video compression algorithms proposed in this thesis are based on zerotree theory, so zerotree theory will be discussed in more detail in Chapter 5.

4.3 Quality Assessment

The quality assessment of compressed visual data can be classified into two categories: subjective quality assessment and objective quality assessment. Subjective quality assessment is performed by observers, while objective quality assessment is calculated by numerical methods.

4.3.1 Subjective Quality Assessment

Since the end results of the compressed visual data are to be viewed by people, it is more appropriate to measure the quality using a subjective quality assessment. Subjective quality measurement is performed by showing images or videos to a number of observers and averaging the results of how good the images or videos appear to them. One example of subjective quality measurement is to use an absolute scale such as the one conducted by Panel 6 (Levels of Picture Quality) of the Television Allocations Study Organization [129]. The scale and the associated definitions are tabulated in Table 4.1.

More recently, subjective quality assessment is formalized in Recommendation ITU-R BT.500-10 [130], which suggests the general test methods, the grading scales, and the viewing conditions, as well as the selection criteria of the test materials and observers. The most commonly used methods are the following:

Number	Name	Description
1	Excellent	The image is of extremely high quality, as good as you could desire.
2	Fine	The image is of high quality providing enjoyable viewing. Interference is perceptible.
3	Passable	The image is of acceptable quality. Interference is not objectionable.
4	Marginal	The image is in poor quality and you wish you could improve it. Interference is some what objectionable.
5	Inferior	The image is very poor but you could watch it. Definitely objectionable interference is present.
6	Unusable	The image is so bad that you could not watch it.

Table 4.1: Subjective Quality Scale used by Panel 6 (Levels of Picture Quality) of the Television Allocations Study Organization [129]

Double Stimulus Impairment Scale (DSIS): The reference signal is shown before the test signal, and both are shown only once. Observers then rate the impairment in the test signal on a discrete five-grade scale ranging from "1" for "very annoying" to "5" for "imperceptible. DSIS is the preferred method when assessing visual signals with clearly visible impairments.

Double Stimulus Continuous Quality Scale (DSCQS): Firstly, the reference and the test signals are consecutively shown to observers one or more times for them to gain a mental measure of both signals. Then, both signals are shown to observers again one or more times for them to rate both signals. Observers are not informed which signal is the reference and which signal is the test signal in both cases. They rate both signals in a continuous quality scale ranging from "1" to "100" for "bad" to "excellent". DSCQS is a reliable assessment scale even when the reference and the test signal are of similar quality.

Single Stimulus Continuous Quality Evaluation (SSCQE): SSCQE is used to assess the time-varying quality of a digitally coded video sequence. The test video sequence which can range from 5 to 60 minutes duration is shown to observers. They continuously rate the instantaneously perceived video quality using a slider on the DSCQS scale. Due to the limitation of the human working memory, the continuously recoded quality results for the entire viewing duration can avoid a biased assessment from observers towards the final 10-20 seconds of a video. In addition, the fact that no reference is used in SSCQE puts observers in a situation close to a real home viewing condition.

Although a subjective assessment can provide a reliable quality rating for compressed visual data, it is a time consuming and resource costly task. Also, a subjective assessment can not be incorporated into a compression scheme to provide an instant quality measurement for predicting the viewing quality or improving the performance. In such cases, an objective quality assessment which can be performed by a computer using numerical methods is a better option.

4.3.2 Objective Quality Assessment

The most widely used and the simplest objective quality assessment is the pixel-based method. It can be determined by measuring either the mean square error (MSE), or the peak signal to noise ratio (PSNR).

The MSE is the mean of the square of the pixel differences between the reference and the test visual signals. For grayscale images, the MSE is defined as

$$MSE(f) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[f(i,j) - \tilde{f}(i,j)]^2, \tag{4.4}$$

where the M and N are respectively the number of rows and columns of the image, and f and $\tilde{f}$ are respectively the reference and the test image. For color images the MSE is defined as

$$MSE = \frac{MSE(R) + MSE(G) + MSE(B)}{3}, \tag{4.5}$$

where R, G, and B are the trichromatic components of the image. For both grayscale and color images, the PSNR is given by

$$PSNR = 10\ log_{10}\frac{255^2}{MSE}. \tag{4.6}$$

The pixel peak amplitude is assumed to be 255, because each pixel in the grayscale image or RGB color components is normally represented by 1 byte. The PSNR provides a logarithmic fidelity scale in decibel (dB) that indicates how closely the test image resembles the reference. Very often the root mean square error (RMSE) is also used to measure the quality. The RMSE is the average difference per pixel, and is given by $RMSE = \sqrt{MSE}$.

When assessing the video quality, the result can represented by single averaged MSE, or PSNR, value by modifying Equation 4.4 to

$$MSE(f) = \frac{1}{TMN}\sum_{k=1}^{T}\sum_{i=1}^{M}\sum_{j=1}^{N}\ [f(k,i,j) - \tilde{f}(k,i,j)]^2, \tag{4.7}$$

where T is the number of frames of the entire video duration. Alternatively, the quality assessment can also be performed by measuring the PSNR of each video frame in a frame-by-frame fashion for the entire duration of the video, just like the SSCQE method discussed previously.

Although MSE, RMSE and PSNR provide a very easy and fast quantitative quality assessment of images, they do not reflect the actual quality perceived by viewers. In some cases, the subjective quality of a image with a lower PSNR can be better than another with a higher PSNR, or two images with similar PSNRs may have a large visual quality difference. Subjective quality critically depends on human visual characteristics and the image or video contents, as well as the viewing conditions, so it cannot be measured by simple pixel-by pixel comparison methods [131].

Human visual characteristics, like the visual processes and the visual insensitivities discussed previously, make some image or video contents are more error resilient than the others. For example, a image with more high spatial frequency contents can tolerate more errors, due to spatial masking; and fast moving video objects can also tolerate more distortions, due to temporal masking. The viewing conditions, like viewing distance and display settings, also influence the perceived quality. For instance, some impairments can not be seen from a large distance but can be seen at a short distance, due to reduction of the spatial frequency leading to an increase of the visual sensitivity. Therefore, in order to mimic human observers in image quality assessment, the human vision characteristics have to be incorporated into the objective quality assessment.

The objective quality measurements that incorporate the human vision characteristics are called *perceptual quality metrics* or *models* [21, 132, 133, 134, 135]. Early quality metrics were based on Schade's single-channel model of the human vision system [46]: The first grayscale image quality metric was developed by Mannos and Sakrison [47, 136]; the first color image metric was developed by Faugeras [137]; and the first video quality metric was developed by Lukas and Budrikis [48].

However, the human visual system is more appropriately modeled in multichannel representation, and so is the quality metric. Some examples are the Visual Differences Predictor (VDP) proposed by Daly [60], the Visual Discrimination Model (VDM) proposed by the Lubin [14], and the *contrast-gain control* model proposed by Teo and Heeger [26]. These metrics all produce a perceptual quality measurement map, which indicates the spatial distortions likely to be detected by human viewers. The VDM was later modified to the Sarnoff *JNDmetrix*^{TM} for color video [138]. The just-noticeable distortion (JND) model, proposed by Jayant [139, 140], gives each signal being represented a threshold of error visibility, below which reconstruction errors are invisible to human observers, ie. visually lossless. In other words, the JND level is the maximum amount of distortion than can be introduced to the image without resulting in any perceived distortions.

In addition, some specialized perceptual quality metrics can also be incorporated to the compression schemes to predict the compression errors or to improve the compression performance. One example for the DCT scheme is DCTune proposed by Watson et al. [141, 142, 27]. The

other examples for the wavelet scheme are the wavelet visible difference predictor (WVDP) model proposed by Bradley [143] and the spatial frequency weighted model proposed by Watson et al. [144].

Although perceptual quality metrics can give a better objective quality assessment, they are too computationally intensive and lack standardization for comparison of results. Also, perceptual quality metrics are designed for assessing images and videos with compression quality at around the JND level. Normally at such quality level, images and video are visually lossless or near-lossless, and have a low compression ratio, ie. high bit-rate. At high compression ratios, the compressed visual data are far beyond the visual threshold, ie. *suprathreshold*, and the perceptual quality metrics simply fail to provide a reliable quality assessment.

Hence, within the scope of this thesis, the quality assessment of the compression results will be the PSNR or the RMSE method, as a *reference scale* only. Whenever possible, the compressed images or video frames will be presented in prints in this thesis, as subjective quality assessment is still the most reliable method in measuring the actual perceived quality.

4.4 Image and Video Compressibility

We have discussed some visual data compression schemes and the methods used to assess the compression results. In this section we will briefly discuss image and video compressibility to gain some ideas about the anticipated results from the compression schemes.

The compressibility of an image or a video is an indicator of how well the image and video can be represented by as little amount of information as possible, but with an acceptable quality. The compressibility mainly depends on the contents in an image or video, regardless of which compression scheme we use. That is, the contents determine the amount of redundant information, and, therefore, how much amount of information can be discarded. The compression schemes then identify and discard the redundant information to achieve compression.

The amount of redundancy depends on the amount of high spatial frequency features in the image, such as edges, contours, texture regions, or leafy areas. If an image has more high frequency features, the image will have lower spatial redundancies, due to lower spatial correlation. Therefore, even after being transformed into the frequency domain, there will still be more high frequency coefficients. In other words, the frequency decomposition cannot decorrelate the image efficiently, so the image has low compressibility. On the other hand, if an image has more low spatial frequency features, like a constant background or a slowly changing background, the image will have more correlation, so that the frequency decomposition will be able to obtain

a more compact data representation. Therefore, the image will have a high compressibility. For video, the compressibility depends on the number of scene changes of moving objects. If there are more scene changes or moving objects there will be less temporal correlation. The video will then have a low compressibility.

In Figure 4.2, we compare the compressibility of two images, *Pot* and *Bird*. The original *Pot* and *Bird* image are shown in Figure 4.2(a) and (b), respectively. From the discussion above, we expect the *Pot* image to have a lower compressibility than the *Bird* image. This is because the *Pot* image has many high frequency features like the plant leaves, the bamboo curtain on the background, and the bamboo veranda with high contrast shadows incident on the curtain. On the other hand, the *Bird* image has mainly low frequency features like the slowly changing clouds and sky on the background, and the low contrast feather texture of the bird. The contour of the bird is the only significant high frequency feature that changes abruptly from one color to another. Figure 4.2(c) and (d) show the frequency decomposition of the Y components of both images by wavelet transform. Because of having more high frequency features, the *Pot* image's wavelet matrix has a significantly higher number of coefficients in the high frequency subbands than the *Bird* image's. Hence, we can anticipate that the *Pot* image will have a lower compression quality than the *Bird* image, if both are compressed at the same ratio. Figure 4.2(e) and (f) show the 100:1 compressed *Pot* and *Bird* image. At such high compression ratio, the *Pot* image loses many high frequency features, due to low spatial correlation; while the *Bird* image looks nearly as good as the original image, due to high spatial correlation.

One thing that must be mentioned here is that in lossless compression, the reduction in the amount of visual data depends mainly on the redundancies but not the visual insensitivities. Only in lossy compression do both redundancies and visual insensitivities have an effect in determining what information will be encoded and what will be discarded.

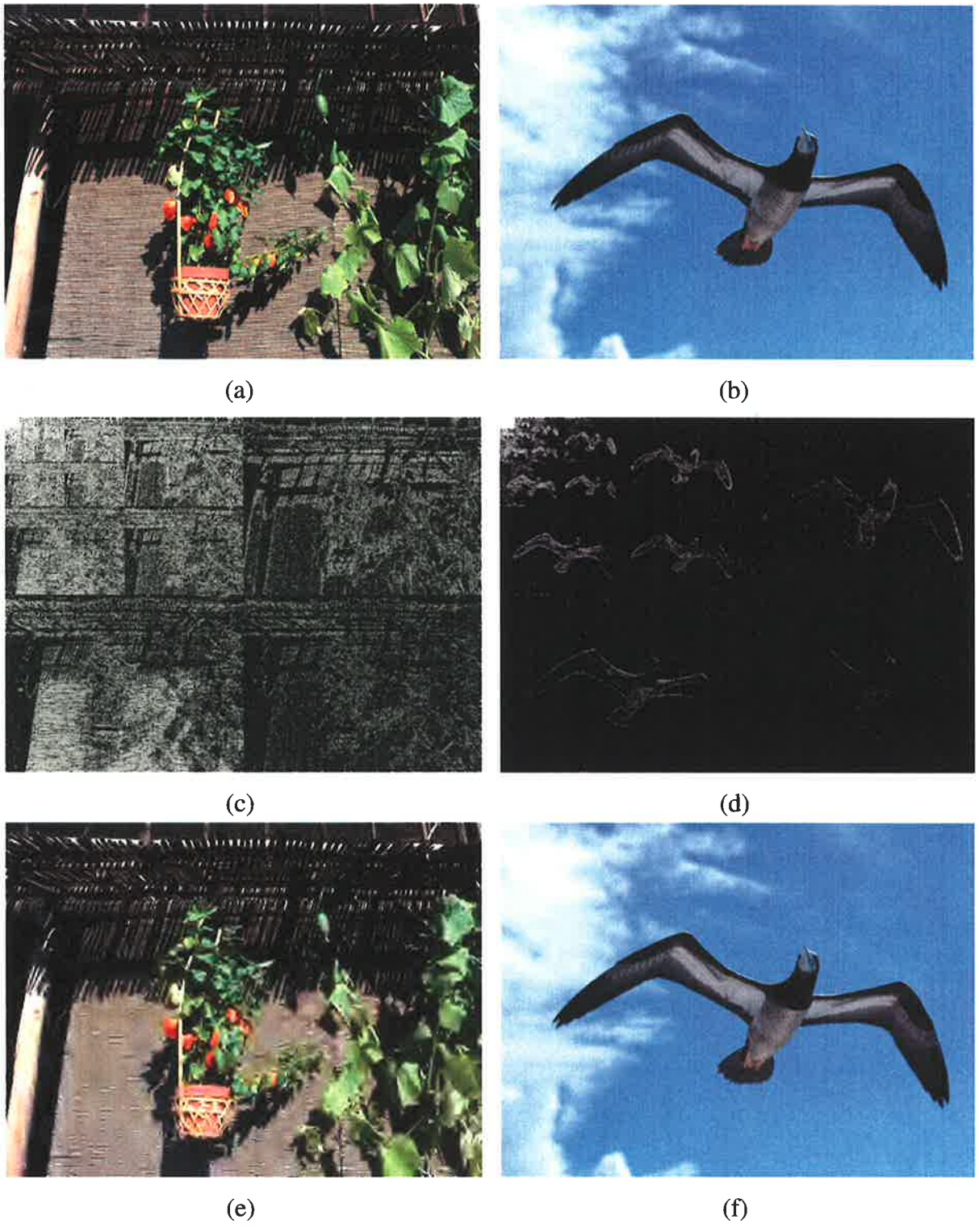


Figure 4.2: Image compressibility comparison (a) the original *Pot* image (b) the original *Bird* image (c) Y component wavelet matrix of *Pot* (d) Y component wavelet matrix of *Bird* (e) 100:1 compressed *Pot* image (f) 100:1 compressed *Bird* Image

Chapter 5

Zerotree Coding Algorithms

5.1 Introduction

As mentioned in the previous chapter, only wavelet compression schemes can achieve acceptable coding quality at low bit-rate applications. Other image and video compression schemes fail to maintain an acceptable coding quality at low bit-rate. Since the late 1980s, a variety of wavelet image compression algorithms have been proposed [145, 146, 147, 148, 101, 149]. These wavelet algorithms have demonstrated a coding performance superior to the JPEG compression standard, especially when the coding bit-rate is low.

However, the major breakthrough for wavelet image compression occurred when Lewis and Knowles invented and implemented the *zerotree data structure*, in their HVS image coding algorithm [126]. The zerotree data structure is significant for two reasons: first, it considers the intra- and inter-subband correlation for identifying the wavelet coefficients that are important to the image quality; second, it results in a very efficient data representation that allows the wavelet coefficients to be represented by a word length lower than their entropy. Later, Shapiro proposed his well-known *embedded zerotree wavelet* (EZW) algorithm [127, 128] based on the zerotree data structure, which strengthened the foundation for zerotree theory to be applied to wavelet image compression.

The significance of EZW is that it requires absolutely no training, no pre-stored tables or code books, and requires no prior knowledge of the image source. Moreover, EZW has several desirable features for image compression and transmission. First, EZW is in the embedded fashion, that is, the coarser scale information is embedded in the finer scale information. This embedded feature is suitable for multi-precision representation for quality scaling. Second, EZW provides an ordered information sequence for transmission, ie. larger coefficients are coded before

smaller ones. This is because, based on information theory, wavelet coefficients with larger magnitude normally contain more information about the original image. Third, EZW offers a very precise rate-distortion control. Since zerotree coding is in the embedded fashion and the important wavelet coefficients are transmitted first, the encoding and decoding procedures can be terminated at any point, if the pre-determined bit-rate or distortion is achieved.

This chapter will discuss zerotree theory in more detail, including the development of zerotree image compression algorithms and the extension of zerotree theory for video compression. This chapter will also discuss the drawbacks of current zerotree image and video compression algorithms. These drawbacks are addressed by the proposed image and video compression algorithms in this thesis.

5.2 Embedded Zerotree Wavelet Coding

Similar to Lewis and Knowles' algorithm, EZW coding exploits the coefficient relationship between the *LL* subband and the high frequency subbands. It is based on the presumption that if a coefficient in the *LL* subband is insignificant with respect to a threshold, the coefficients of the same spatial location in the high frequency subbands also will have a high probability of being insignificant. Therefore, wavelet coefficients of the same spatial location can be grouped into a tree structure, and the significance of this tree can be indicated by the *root* coefficient in the *LL* subband.

The parent-child dependency of the wavelet coefficient tree is shown in Figure 5.1(a). The coefficient in the *LL* subband is the root (or parent) coefficient, which has three *child* coefficients in the higher frequency subbands of the same transform level. Each of these *child* coefficients has four children in the same spatial locations of the finer transform level. Coefficients in the finest (or lowest) transform level are the *leaf* coefficients, which do not have any child coefficient. The wavelet matrix in Figure 5.1(a) has three transform levels, so one *root* coefficient will have a total of 63 descendants spanning three generations.

EZW coding has two coding passes, namely the *dominant pass* and the *subordinate pass*. The wavelet coefficients are scanned in the raster order shown in Figure 5.1(b). The scanning order is in a fashion such that no child coefficient is scanned before its parent.

In the dominant pass, the magnitudes of wavelet coefficients are compared with a threshold T. The initial threshold T_0 is chosen such that $x_{ij} \leq 2T_0$ for all wavelet coefficients x_{ij}. The threshold is halved in every subsequent dominant pass. This results in a *bit-layer* encoding as illustrated in Figure 5.2. The coefficients of a wavelet transform matrix are separated into a sign

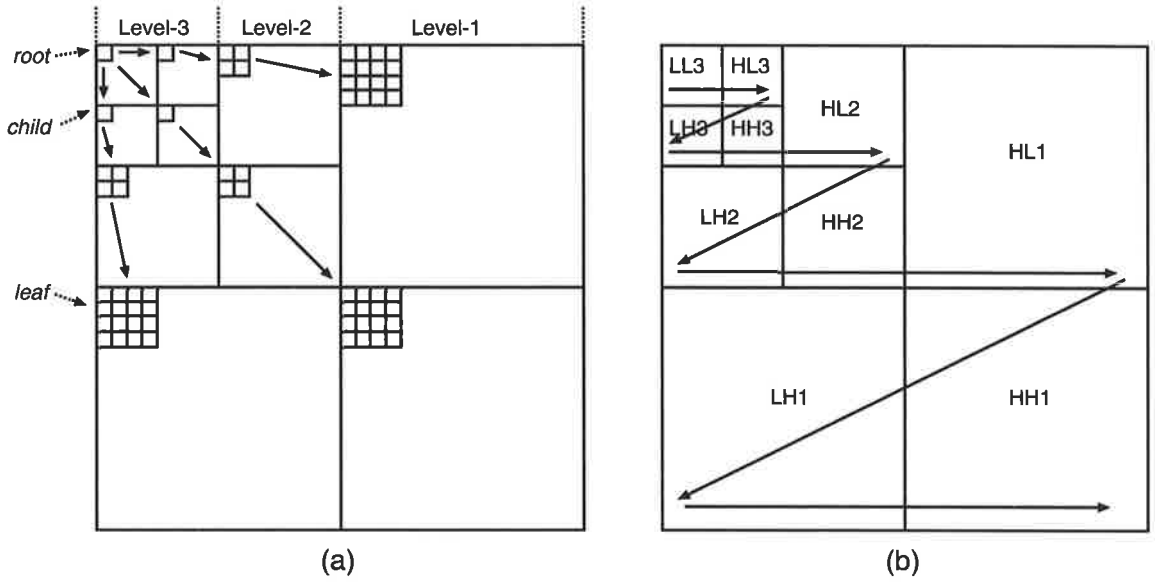


Figure 5.1: (a) EZW coefficient parent-child dependency (b) EZW tree scanned order

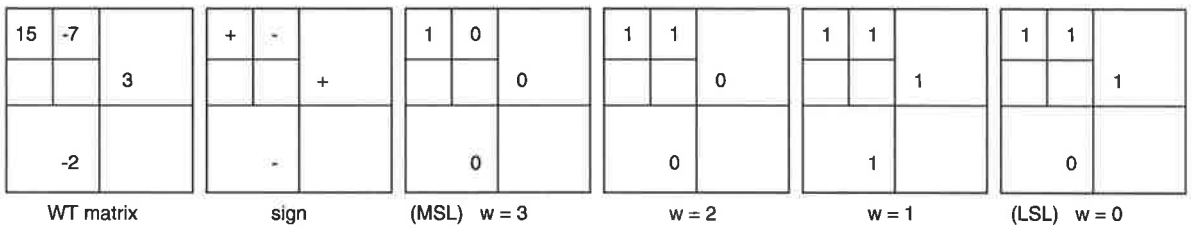


Figure 5.2: Bit-layer coding diagram

layer and several magnitude layers. The *weight* of a bit-layer is indicated by w. A coefficient whose most significant bit (MSB) appears in the bit-layer with weight w means its magnitude falls between 2^w and $2^{w+1} - 1$. The layer coding sequence is from the *most significant layer* (MSL) to the *least significant layer* (LSL).

As discussed in the previous chapter, the distortion measurement of the reconstructed image is given by the MSE,

$$D_{mse}(F - \tilde{F}) = \frac{||F - \tilde{F}||^2}{MN} = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} [f(i,j) - \tilde{f}(i,j)]^2, \tag{5.1}$$

where F is the original image and $\tilde{F}$ is the reconstructed image. If the wavelet transform is a unitary transform, Equation (5.1) can be written as

$$D_{mse}(F - \tilde{F}) = D_{mse}(C - \tilde{C}) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} [c(i,j) - \tilde{c}(i,j)]^2, \tag{5.2}$$

where C and $\tilde{C}$ are the original and reconstructed wavelet matrix, and $c(i,j)$ and $\tilde{c}(i,j)$ are their coefficients at coordinate (i,j). If a wavelet coefficient $c(i,j)$ is sent to the decoder, the distortion, D_{mse}, will be reduced by $|c(i,j)|^2/MN$. Intuitively, larger coefficients are more important for reducing the reconstructed error and should be transmitted first. Therefore, EZW begins encoding from the MSL to ensure that a reasonable reconstructed image quality can be obtained earlier (information ordering), and also to ensure that the coding process can be terminated at any point (embeddedness).

Depending on the magnitude and the sign, a coefficient is coded into four different symbols, namely positive coefficient "+", negative coefficient "-", zerotree root *"ZTR"*, and isolated zerotree *"IZ"*. If the magnitude of a coefficient is greater than the current threshold, that coefficient is coded as symbol "+" or "-" depending on its sign. On the other hand, if the coefficient and its descendants are all below the threshold, this coefficient is coded as a *"ZTR"*. However, if the coefficient is below the threshold but one or more of its descendants are greater than the threshold, this coefficients is coded as an *"IZ"*. For those coefficients that are coded as "+" or "-", their magnitudes are added to the *magnitude list* and are set to zero in the wavelet matrix.

The subordinate pass performs two tasks. Firstly, it compares the values of new entries in the *magnitude list* with an uncertainty interval which is between T and $2T$, where T is the current threshold. If the value is in the upper half of the uncertainty interval, it will be coded as symbol *"1"*, whereas if the value is in the lower half of the uncertainty interval, it will be coded as symbol *"0"*. Those symbols are then added to the *subordinate list*. Secondly, the subordinate pass refines the coefficients already in the *magnitude list* by comparing them with the new uncertainty interval. The output symbols are the refinement symbols for the coefficients in the *magnitude list*.

In the encoding process, the position information is encoded by the zerotree symbols, "+", "-", *"ZTR"*, and *"IZ"*, while the magnitude information is encoded by the subordinate symbols, *"1"* and *"0"*. Therefore, the EZW decoder can just use these two types of symbols to reconstruct an approximate version of the original wavelet matrix.

5.3 Set Partitioning In Hierarchical Trees

Because EZW has demonstrated the efficiency of the zerotree data structure for image compression, several other algorithms have been developed based on zerotree theory. Some examples are Tsai et al.'s Stack-Run image coding [150], Algazi and Esters' Analysis Coding [151], RICOH's CREW [152, 153], Davis' wavelet-based Fractal Image Compression [154], Creusere's REZW [155], and Said and Pearlman's Set Partitioning in Hierarchical Trees

(SPIHT) [156]. Although zerotree compression performance has been improved by those algorithms, only SPIHT concentrates on providing a simpler and faster coding method.

Compared to EZW, SPIHT has several advantages. First, a SPIHT coder searches the tree recursively, ie. one tree branch is dealt with at once, therefore the coder requires less temporary memory to store the information of the parent-child relations. Second, unlike the four zerotree symbols in EZW, SPIHT uses only two symbols to represent position information. Consequently, the bit-budget can be used for coding more coefficients, rather than coding position information. Third, thresholds in SPIHT are always chosen to be power of two. Therefore, comparing a coefficient with the threshold is just a logical *AND* operation, and a reconstructed value is just the summation of several *dyadic numbers*. That is, SPIHT uses integer operations, which are more hardware implementation friendly. Finally, unlike EZW, a SPIHT coder can search, compare, and encode wavelet coefficients with a single coding pass. Therefore, the coding complexity of SPIHT is less than that of EZW.

A SPIHT coder maintains three coding lists. These three lists are a list of insignificant pixels (LIP), a list of significant pixels (LSP), and a list of insignificant sets (LIS). These three lists are used to store the coordinates of significant coefficients, insignificant coefficients, and insignificant tree sets, respectively. Like an EZW coder, a SPIHT coder also performs two coding passes. The first coding pass is the *sorting pass* that sorts tree branches into hierarchical order, adds the coordinates to coding lists, and outputs the magnitude and sign information of coefficients. The second pass is the *refining pass* that outputs refinement bits for the coefficients whose coordinates are stored in the LSP.

The parent-child relation of SPIHT is shown in figure 5.3(a). This relation is different from that of EZW. Instead of forming individual tree roots in the LL subband, SPIHT groups coefficients in the LL subband into 2×2 groups. In this 2×2 group, one coefficient is an *isolated tree root*, indicated by "*", and each of the other three roots has four children in the high frequency subbands. With this grouping, a SPIHT coder can deal with more coefficients at once than EZW for the same number of wavelet transform levels. Said and Pearlman named this tree structure the *spatial orientation tree*. Figure 5.3(b) shows an example of the recursive tree searching performed by SPIHT. The numbers marked in the tree coefficients indicate the visiting order by the SPIHT coder when that branch is coded.

The decoding process of SPIHT is simply the reverse of the encoding process. In addition, because SPIHT inherits all of the advantages of EZW, the encoding and decoding processes can also be terminated at any point, giving a precise rate or distortion control.

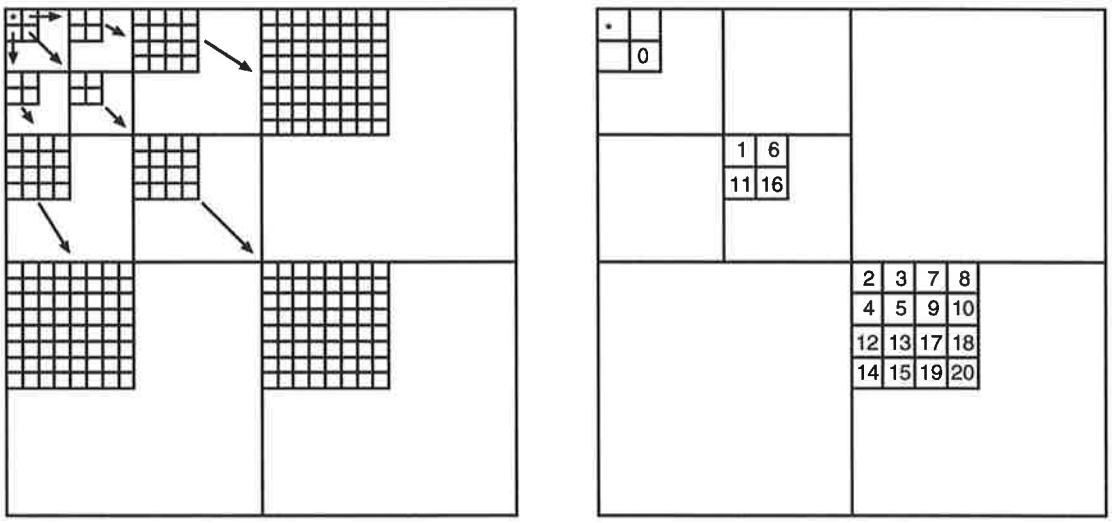


Figure 5.3: (a) SPIHT zerotree (hierarchical tree) parent-child dependency (b) SPIHT recursive tree searching order

5.4 Three Dimensional Wavelet Zerotree Video Coding

Since zerotree coding has a superior coding performance to DCT coding, researchers have also applied zerotree theory in video coding. A video sequence is an array of frames, and normally contains high temporal correlation, which results in significant information redundancy. Therefore, in order to achieve a better compression efficiency, this temporal correlation will have to be removed before applying any spatial coding to each video frame. The method used by the international standards, such as MPEG and H.263, to remove the temporal correlation is called *motion estimation and compensation* (ME/C). The combination of ME/C and zerotree theory for video coding has been demonstrated by Wang and Ghanbari [157], Martucci et al. [158], and Pearlman et al. [159].

Alternatively, the removal of temporal correlation can also be done by the three dimensional (3D) wavelet transform, which is an extension of the 2D wavelet transform to the temporal domain. The exploitation of the 3D wavelet transform for subband video coding was demonstrated by Karlsson and Vetterli [160], Taubman and Zakhor [161], Ohm [162], and Levy and Wilson [122, 123]. Figure 5.4 gives an example of the application of the 3D wavelet transform to a video sequence. Before transformation, the video frames will have to be grouped in a group of frames (GOF) of dyadic size. In this example, the size of the GOF is four frames. The 3D wavelet transform decomposition of this GOF is first performed along the temporal axis for temporal decomposition, and then in the spatial domain for spatial decomposition. After several applications of the temporal wavelet transform, a GOF will have the temporal subband orien-

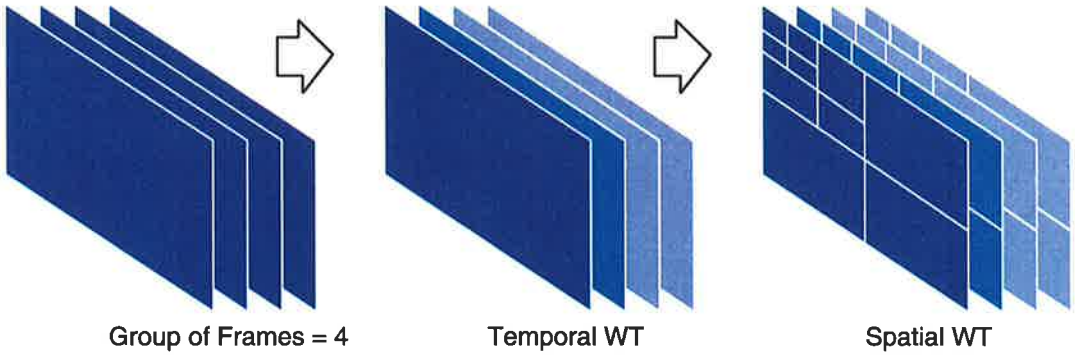


Figure 5.4: 3D wavelet transform applied to a group of video frames

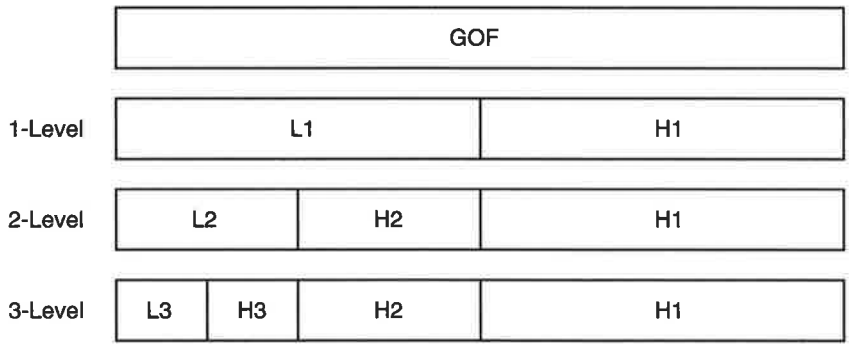


Figure 5.5: Subband orientation of a GOF after being 3-level temporal wavelet decomposition

tation shown in Figure 5.5. Letters *L* and *H* denote low-pass and high-pass temporal wavelet transforms, respectively. Because of a strong inter-frame correlation, the information in the GOF will be mostly packed into the temporally lowest frequency subband, *L3*; while the high frequency temporal subbands will sparsely contain the high temporal frequency information corresponding to the changes within the GOF.

The 3D wavelet transform generates coefficients in a *3D hierarchical subband structure*, which is ideal for zerotree coding. Interestingly, despite the excellent performance on still image compression, only a few algorithms have ever combined zerotree theory and 3D wavelet transform for video coding. Those algorithms are HVS video coding of Lewis and Knowles [163], Tri-Zerotrees of Tham et al. [164, 165], and 3D SPIHT of Kim et al. [159, 166, 167, 168]. Among these algorithms, 3D SPIHT gives the best compression results and has the lowest coding complexity. Therefore, 3D SPIHT is the current benchmark for 3D wavelet zerotree video coding algorithms.

The zerotree structure of 3D SPIHT is called a *spatio-temporal orientation tree*. One example is shown in Figure 5.6. The tree roots in the lowest frequency subband are marked by *,

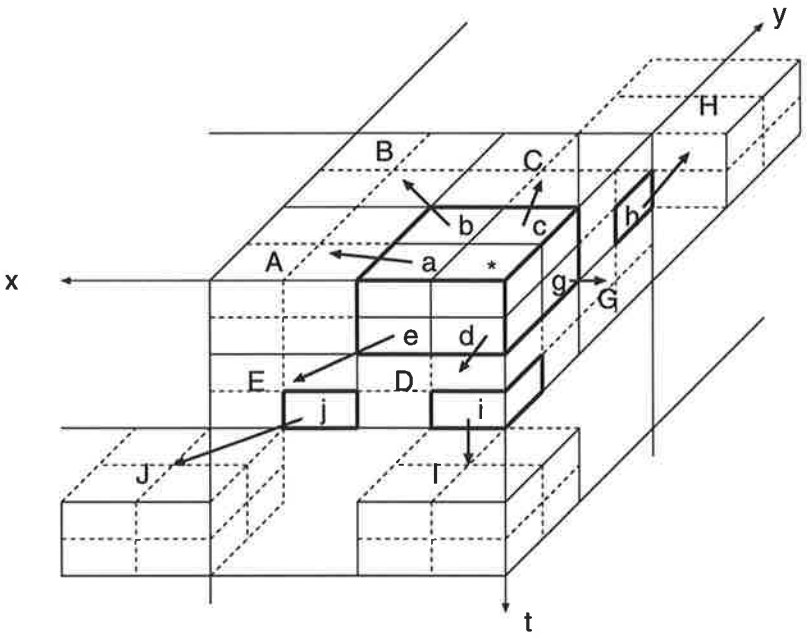


Figure 5.6: 3D SPIHT zerotree (hierarchical tree) parent-child dependency

a, b, c, d, e, f, and *g*, noting that *f* is hidden below *b*. Their child sets, except for the isolated tree root *, are marked by *A, B, C, D, E, F*, and *G*, where *F* is hidden below *B*. Figure 5.6 also shows the tree nodes, *h, i*, and *j*, and their associated child sets, *H, I*, and *J* in the high frequency subbands. Each parent node has eight children, which form a 2x2x2 coefficient block in the same spatio-temporal location in the finer transform level.

Since 3D SPIHT is the extension of SPIHT to the spatio-temporal domain, a 3D SPIHT coder also maintains three coding lists to store the sorting results along the spatio-temporal orientation trees. In addition, the 3D SPIHT coder has a sorting pass and a refining pass. The sorting pass recursively sorts the coefficients in a tree branch according to their magnitudes with respect to the coding threshold. The sorting results are stored in those three coding lists. After that, the refining pass refines the magnitude information for those coefficients with their coordinates already being stored in the LSP (list of significant pixels).

Compared to ME/C video coding, 3D wavelet video coding has several advantages. First, 3D wavelet video coding has a lower computational complexity, because it exploits the fact that the temporal correlation can be decorrelated after a very simple temporal wavelet transform, due to the information packing property of the wavelet transform. In contrast, ME/C video coding will generally require a very intensive pixel-based search among the successive video frames to identify and remove temporal redundancy. Second, since ME/C is a predicting and compensating process, its regularity is highly dependent on the video contents. On the other hand, the 3D wavelet transform is a very regular process, regardless of the video contents.

Third, the 3D wavelet transform produces the transformed GOFs with a hierarchical coefficient structure. Therefore, 3D wavelet video coding can easily produce a multiresolution bit-stream for video quality scaling that enables a wider number of applications. On the other hand, ME/C video coding can only produce a video bit-stream with a fixed quality, which is limited to a certain number of applications. Finally, 3D wavelet video coding is not only suitable for realtime software implementation but also is suitable for hardware implementation, because of its simplicity and high regularity.

5.5 Drawbacks of Zerotree Coding Algorithms

Since SPIHT has shown a very efficient way to compress still images, many ad hoc image compression algorithms based on it have been proposed, such as [169, 170, 171, 172, 173]. All of the zerotree coding algorithms have three distinctive features: they require coding lists, they perform multiple coding passes, and they use multiple zerotree symbols. However, these distinctive features are also the drawbacks of zerotree coding algorithms. They increase the coding memory requirement and the coding complexity.

Firstly, zerotree coding algorithms require some coding lists to store coding information; two coding lists for EZW and three coding lists for SPIHT. These coding lists are essential for the formation of the embedded coding. However, the use of coding lists will demand a significantly large amount of coding memory. For example, when using SPIHT to code a grayscale 512x512 image, one single entry of the list requires 18 bits or 2.25 bytes of memory to store the row and column coordinates. Given that the total number of list entries is approximately twice the total number of coefficients, the total memory required will be 1.125 MB [1]. When coding a color image of the same size, each color component will require its own coding lists, and this increases the memory requirement to 3.375 MB. The coding memory requirement will increase even more when increasing the target coding bit-rates or image sizes.

Similarly, 3D SPIHT also requires a large amount of coding memory to maintain its three coding lists. For example, when coding a color (4:1:1) QCIF video sequence with 16 frames in a GOF, a single list entry will require 20 bits of memory for the Y component and 18 bits of memory for the U and V components to store the 3D coordinates. Suppose at a moderate bit-rate, the total number of list entries is the same as the total number of pixels, the memory required by 3D SPIHT will be about 1.4MB [2]. However, when increasing the video frame sizes to (4:1:1) CIF format, the memory requirement will increase to about 6.19 MB, not to mention

[1] 2.25(bytes)x512x512x2/1024/1024 = 1.125 MB

[2] [176x144x20(bits) + 88x72x18(bits)x2(colors)] x 16(GOF)/8/1024 = 1.4 MB

the 4CIF applications.

In spite of having high coding efficiency, the high coding memory requirement makes zerotree coding algorithms, such as SPIHT and 3D SPIHT, not suitable for hardware implementation. In particular, they are not suitable for portable devices, as the minimization of memory is one of the important design issues in reducing device size and power consumption.

Secondly, zerotree coding algorithms normally need to perform multiple coding passes, like the sorting and refining passes performed by SPIHT and 3D SPIHT. Those multiple coding passes are also the key processes for maintaining the embeddedness of the coded bit-stream. The multiple coding passes can be easily done by a software implementation but not by a hardware implementation. For the hardware implementation, switching between the multiple coding passes will require buffering, which consequently increases the implementation complexity and memory requirement. Therefore, the hardware implementation of multiple coding passes will be very costly. Furthermore, since 3D wavelet video coding algorithms generally suffer from the problem of coding latency due to the formation of GOF, the multiple coding passes performed by 3D SPIHT would cause a real problem in interactive applications such as videophones.

In addition to multiple coding passes, the use of multiple symbols in zerotree coding algorithms will also increase the coding complexity. This is because those symbols need to be encoded by bit-budget, so that they can be distinguished between coder and decoders. Therefore, the coding efficiency may be degraded if too many coding symbols are used.

Therefore, in order to reduce the memory requirement and the coding complexity, the coding lists must be removed, and the number of coding symbols used must be minimized. Some new zerotree algorithms have been proposed to reduce the memory requirement problem, such as the low-memory packetized SPIHT [174] and the no list SPIHT [175], both proposed by Wheeler and Pearlman. However, some lists are still used in these two coding algorithms. The total removal of coding lists has been attempted by Su and Wu in their Embedded Recursive Zerotree (ERZ) coding scheme [176]. However, because of the failure to use the inter-subband relations between the LL and other high frequency subbands in the zerotree data structure, and the redundant assignment of wavelet coefficients to zerotree symbols, ERZ has a limited coding performance.

Apart from Lewis and Knowles' algorithm, the current zerotree algorithms are only optimized for the reduction of the MSE or PSNR, but not for the human vision system. In Lewis and Knowles' algorithm, the characteristics of the HVS are taken into account for determining the quantization thresholds [177]. Those characteristics include luminance adaption, spatial masking, frequency band sensitivity, and texture masking. As discussed previously in Chapter 2, those characteristics of the HVS have a direct influence on the actual perceived quality of

the compressed image and should be considered in the compression algorithms. Ignoring the characteristics of the HVS may cause those zerotree algorithms to have a good performance in terms of MSE and PSNR only, but not in terms of perceived quality.

In this thesis, a new zerotree algorithm for color image compression is proposed. It reduces the drawbacks of the other zerotree algorithms discussed above. Since there is *no* list used in the proposed algorithm, it is named *Listless Zerotree Coding* (LZC). When compared to other zerotree coding algorithms, LZC has three advantages. First, because it uses no coding list, the coding memory requirement is significantly reduced. The memory requirement of LZC is fixed for all bit-rates, and is only a fraction of that required by SPIHT. It is only proportional to the image size. Second, the multiple coding passes normally performed by other zerotree algorithms have been reduced to just one pass, which results in a significant reduction in the coding complexity. Third, LZC uses only two zerotree symbols to code the wavelet coefficients, so there is also a coding complexity improvement. Beside these improvements, LZC also considers the characteristics of the HVS in its bit-allocation process, trying to improve the perceived quality of the reconstructed images.

Since only few 3D zerotree video algorithms have ever been proposed, the LZC algorithm is also extended to video coding by combining it with the 3D wavelet transform. This novel video coding algorithm is called 3DLZC. 3DLZC not only provides an alternative 3D zerotree video coding algorithm, but also dramatically reduces the large coding memory requirement and the coding complexity of 3D SPIHT. We will discuss LZC and 3DLZC in detail in Chapter 6 and Chapter 7, respectively.

Chapter 6

Listless Zerotree Coding for Color Image Compression

6.1 Introduction

In this chapter we will present a zerotree coding algorithm called Listless Zerotree Coding (LZC) that requires no coding lists, resulting in a very low memory requirement. The multiple coding passes performed by other zerotree algorithms have been reduced to just one pass in LZC, resulting in a lower coding complexity.

We will first discuss how we can take into account the characteristics of the human visual system in the LZC algorithm. Then, we will discuss in detail the topics related to the LZC algorithm, including LZC tree symbols and structure, significant maps, tree searching methods, and coding procedures. After that, we will discuss some implementation issues and show some preliminary testing results from LZC test codecs. Finally, we will compare the LZC algorithm with SPIHT and the JPEG 2000 standard.

6.2 Mapping HVS Characteristics in a Bit-Allocation Scheme

In Chapter 2 and 3, we discussed the human visual system characteristics that are closely related to image compression. Therefore, in this section we will discuss how we can map those characteristics in the bit-allocation scheme used by the LZC algorithm.

The considerations of the HVS characteristics in the bit-allocation scheme can be separated into two aspects: within a wavelet matrix and among color components.

6.2.1 Bit-Allocation Within a Wavelet Matrix

From the discussion in Section 3.3 we have seen that the wavelet decomposition is similar to the cortex decomposition of our visual system. This similarity enables us to identify which part of the wavelet matrix contains more important information for the visual system, and which part does not. To locate which part of the wavelet matrix contains more important information than the other, we can first look at the relationship between the spatial frequency decomposition property of the wavelet transform and the contrast sensitivity characteristic of our visual system.

After a wavelet transform, each subband will contain information within a certain horizontal and vertical range, depending on the transform level. The coarser the transform level, the lower the spatial frequency. From the discussion in Section 2.3.2 we know that our visual system has a higher contrast sensitivity toward the low spatial frequency visual signals, but has a lower contrast sensitivity toward the high spatial frequency visual signals. Therefore, the information in the lower frequency subbands is more important than that in the high frequency subbands.

The horizontal spatial frequency range of a subband is given by

$$F_H(l) = \begin{cases} [0, 2^{-l} f_{maxH}] & \text{if low-pass filtered ,} \\ [2^{-l} f_{maxH}, 2^{-l+1} f_{maxH}] & \text{if high-pass filtered.} \end{cases} \tag{6.1}$$

Similarly, the vertical spatial frequency range of a subband is given by

$$F_V(l) = \begin{cases} [0, 2^{-l} f_{maxV}] & \text{if low-pass filtered ,} \\ [2^{-l} f_{maxV}, 2^{-l+1} f_{maxV}] & \text{if high-pass filtered.} \end{cases} \tag{6.2}$$

The parameter l in both equations denotes the transform level of the subband, and f_{max_H} and f_{max_V} are the maximum horizontal and vertical spatial frequencies, respectively. They are given by

$$f_{maxH} = \frac{R_W}{2\alpha_H} \quad \text{and} \quad \alpha_H = 2tan^{-1}\frac{W}{2D}, \tag{6.3}$$

$$f_{maxV} = \frac{R_H}{2\alpha_V} \quad \text{and} \quad \alpha_V = 2tan^{-1}\frac{H}{2D}, \tag{6.4}$$

where R_W and R_H are the image horizontal and vertical dimensions in $pixels$, α_H and α_V are the horizontal and vertical visual angles in $degrees$, W and H are the image width and height in cm, and D is the viewing distance in cm.

For instance, the *HL2* subband in Figure 6.1 contains information from the horizontal range of $0.25 f_{maxH}$ to $0.5 f_{maxH}$ and the vertical range of 0 to $0.25 f_{maxV}$. After knowing the subband frequency ranges, we can then map the CSF of a certain viewing condition to the wavelet subband and interpret the importance of the subband information for that viewing condition, such as the mapping of the *HL2* subband shown in Figure 6.1.

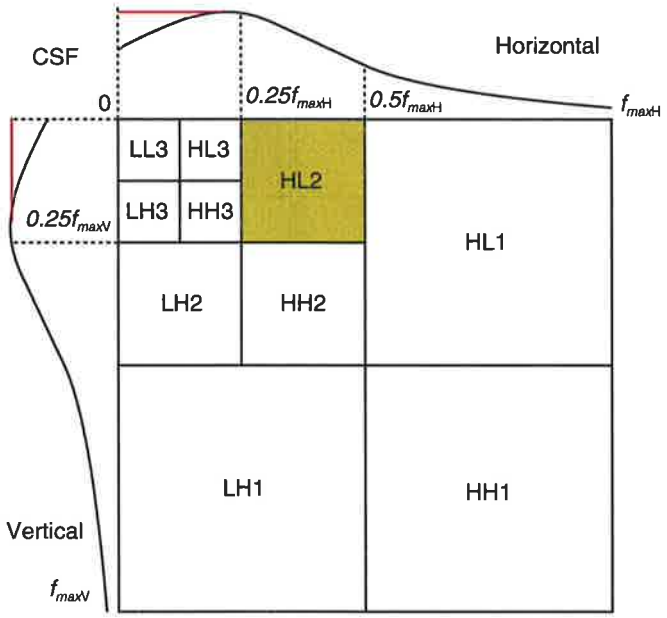


Figure 6.1: Spatial frequency ranges of wavelet subbands

If the frequency ranges of a subband correspond to a higher CSF, that subband will contain more visually important information and require more bit-budget. On the other hand, if the frequency ranges correspond to a lower CSF, that subband will contain less important information and requires less bit-budget, because it is more visually error resilient. For instance, in Figure 6.1, *HL3*, *LH3* and *HH3* subband are more important than the others, so a bit-allocation scheme should allocation more bit-budget to those subbands. However, from Equation (6.3) and (6.4) we can see that if we increase the viewing distance, the spatial frequency ranges of *LL3* subband will be extended to include the high CSF range. Therefore, as discussed in Section 2.3.2, the CSF is normally flattened in the low frequency range, as indicated by red lines in Figure 6.1. The *LL3* subband should then be allocated the same amount of coding bit-budget as *HL3*, *LH3* and *HH3* subband. As a result, the bit-allocation sequence should begin at lower frequency subbands and end at higher frequency subbands, ie. from the coarsest transform level to the finest transform level.

Besides the frequency decomposition property, the wavelet transform also has the orientation decomposition property. After a wavelet transform, the image orientation information will be decomposed into three orientations. As discussed in Section 3.3.3, we know that *LH* subbands contain horizontal high frequency details, *HL* subbands contain vertical high frequency details, and *HH* subbands contain diagonal high frequency details. The importance of the information in *LH*, *HL*, and *HH* subbands can then be interpreted by looking at the orientation sensitivity of our visual system, as discussed in Section 2.3.3. Due to the oblique effect, our visual system is more sensitive to the signals in the horizontal and vertical orientations, but less sensitive to the signals

in the diagonal orientation. That is, *LH* and *HL* subbands contain more important information than *HH* subbands. Therefore, a bit-allocation scheme should allocate more coding bit-budget to code wavelet coefficients in *LH* and *HL* subbands than those in *HH* subbands. Consequently, the bit-allocation sequence for coding orientation information should be *LH*→*HL*→*HH*.

Figure 6.2 gives an example of how a bit-allocation scheme can influence the visual quality of the reconstructed image. Figure 6.2(a) shows 2-level transformed wavelet matrix of *Lena* image. Figure 6.2(b) shows the reconstructed version by using only *LL* subband coefficients without considering the characteristics of our visual system. The compression ratio is 16:1. However, since our vision is also sensitive to the high spatial frequency information contained in the high frequency subbands, omitting high frequency information will cause the reconstructed image to look blurry. On the other hand, Figure 6.2(c) shows the reconstructed *Lena* image which was compressed to the same compression ratio using the LZC algorithm. Since the bit-allocation scheme in LZC algorithm follows the characteristics of our visual system, the *Lena* image looks sharper and more visually pleasant.

Figure 6.2: Bit-allocation scheme comparison on the *Lena* image (a) 2-level wavelet transform matrix (b) using only *LL* subband coefficients (c) using LZC algorithm

6.2.2 Bit-Allocation Among Color Components

For color image compression, the image is normally transformed from RGB color space to one of the luminance-chrominance color spaces, mostly to YCbCr or YUV. After the color space transform, we can obtain a more compact color data representation, that is similar to the visual signal representation in the opponent color channels of our visual system. Based on the discussion in Section 3.2 we know that for YCbCr and YUV color spaces, the Y component

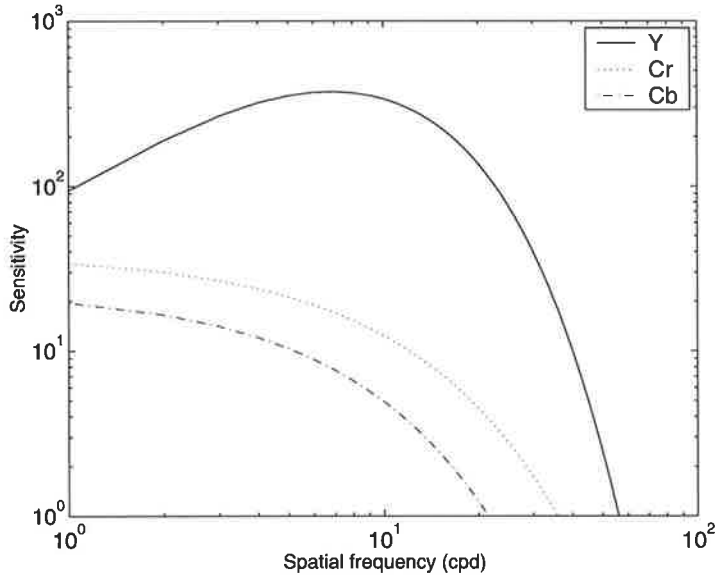


Figure 6.3: Example of color component CSFs - YCbCr color space

is analogous to B-W channel, Cb and U are analogous to the B-Y channel, and the Cr and V components are analogous to the R-G channel. Therefore, we can determine the importance of the information contained in those color components by looking at the color contrast sensitivity of our visual system

From the discussion in Section 2.5.2, we know that the contrast sensitivity of opponent color channels varies along the perceivable spatial frequency range. Among all three opponent color channels, the B-W channel has the highest contrast sensitivity, as its CSF is the highest and spans the entire range of the perceivable spatial frequency. On the other hand, B-Y and R-G channels have relatively lower contrast sensitivity, and their CSFs decline rapidly at moderate spatial frequency. The CSF of B-Y channel especially starts to decline at low spatial frequency. As a result, we expect to see the color components of YCbCr and YUV color spaces have similar contrast sensitivity behavior to the opponent color channels.

Figure 6.3 shows one example of the CSFs of YCbCr color spaces from [178]. Clearly, the CSF of the Y component is higher than the CFSs of Cb and Cr components. We also can see that the CSF of Cb is the lowest and starts to decline at around 8 cpd. The CSFs of the YUV color space is not shown, but they are very similar to those of YCbCr, as the two color spaces are closely related.

Therefore, from the discussion above, signals in the Y component are the most important to our vision, so the bit-allocation scheme should allocate more coding bit-budget for them to ensure a better reconstructed visual quality. Moreover, signals in chrominance components are

not as critical as those in the Y component for reconstructed quality, so chrominance components will not receive as much bit-budget as the Y component and will be quantized coarsely. However, the bit-allocation scheme should still ensure that Cr and V components get more bit-budget than Cb and U components. Hence, the color coding sequence for a source coding scheme should be Y→Cr→Cb, or Y→V→U, depending on which color space used.

6.3 LZC Zerotree

This section gives an overview of the zerotree symbols, the zerotree structure, and the significants maps used in the LZC algorithm.

6.3.1 LZC Zerotree Symbols

One of the distinctive features of zerotree coding algorithms is the assignment of wavelet coefficients to a set of zerotree symbols. The function of those zerotree symbols is to indicate the positions of wavelet coefficients. The zerotree symbols may represent an individual coefficient or a group of coefficients.

The zerotree symbols used in the LZC algorithm are as follows:

- R- root coefficient set in LL subband;
- $C(i,j)$- wavelet coefficient at coordinate (i,j);
- $O(i,j)$- child coefficient set of $C(i,j)$;
- $D(i,j)$- descendant coefficient set of $C(i,j)$.

If $C(i,j) \in R$, then $O(i,j) = [(i+m_r, j), (i, j+n_r), (i+m_r, j+n_r)]$, where m_r and n_r are respectively the number of rows and columns of the LL subband. On the other hand, If $C(i,j) \notin R$, then $O(i,j) = [(2i, 2j), (2i, 2j+1), (2i+1, 2j), (2i+1, 2j+1)]$. Symbols R, $O(i,j)$, and $D(i,j)$ all represent a group of coefficients. Parameters i and j are the row and column index, respectively. The functions of $O(i,j)$ and $D(i,j)$ are to specify the starting coordinates for locating child or descendant coefficients.

Although the LZC algorithm assigns wavelet coefficients to four zerotree symbols, the actual positions of wavelet coefficients are only coded by symbols $C(i,j)$ and $D(i,j)$. Symbols R and $O(i,j)$ are only used as position references in the encoding and decoding procedures. In a later section, we will discuss more about the wavelet coefficient coding using these zerotree symbols.

6.3.2 LZC Zerotree Structure

Another distinctive feature of the zerotree coding algorithm is that wavelet coefficients with inter-subband correlations are grouped into a wavelet coefficient tree. There are two benefits for doing so. Firstly, the significance of a group of wavelet coefficients can be effectively coded using only a small amount of coding bit-budget. This is because wavelet coefficients with inter-subband correlations represent the information from the same spatial location in an image, and have similar magnitude significance with respect to a threshold. Secondly, the location information of wavelet coefficients can also be very efficiently coded with a small amount of bit-budget, as the location information is already embedded in the tree structure.

The LZC algorithm uses a tree structure similar to the EZW algorithm to group wavelet coefficients. Each highest level coefficient tree has a *root* coefficient in the *LL* subband, and has three tree branches in *HL*, *LH*, and *HH* orientations. Figure 6.4 shows an example of a LZC tree structure on a 3-level transformed wavelet matrix, together with the zerotree symbol assignment to one of the tree branches. A coefficient in the *HH3* subband is assigned to symbol $C(i,j)$, and its four children in the *HH2* subband are assigned to symbol $O(i,j)$. The entire descendant coefficient set of $C(i,j)$ is assigned to $D(i,j)$ and is enclosed by the circle in the figure. According to the assumption used in zerotree coding, if $C(i,j)$ is smaller than the coding threshold, its descendant set $D(i,j)$ is also likely to be smaller than the coding threshold. If this is the case, $D(i,j)$ will then be coded as a *zerotree*.

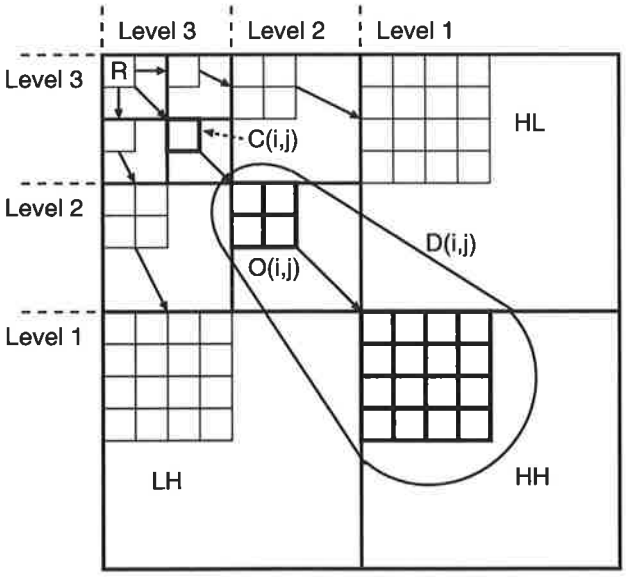


Figure 6.4: LZC wavelet coefficient tree structure

The reason for using a zerotree structure similar to EZW is that there will be more flexibility in the choice of image dimension. The zerotree structure of SPIHT requires coefficients in the *LL* subband to be grouped into a 2×2 block, as shown in Section 5.3. However, this 2×2 grouping is applicable when the image has a large size, but not a small size. This is because grouping coefficients into a 2×2 block requires an even dimension for the *LL* subband, but when the image size is small it is likely there will be only a few dyadic wavelet transform levels available, so it is unlikely to maintain an even dimension for the *LL* subband. In contrast, the EZW zerotree structure has only one coefficient as a root in the *LL* subband. Therefore, the dimension of the *LL* subband can be either even or odd, leaving a lot more flexibility in the choice of both image size and number of wavelet decomposition levels.

6.3.3 LZC Significance Maps

In order to have an embedded bit-stream, a zerotree algorithm normally has to sort the wavelet coefficients by their magnitudes. Therefore, it is essential to maintain a set of coordinate lists for storing the original locations of wavelet coefficients and coefficient sets. However, the necessity to maintain a set of coding lists is the significant drawback for other zerotree algorithms, as a large amount of coding memory will be required.

The LZC algorithm eliminates the need for maintaining coding lists by using a set of significance maps instead. Because only two zerotree symbols are used to code the position information of wavelet coefficients, the LZC algorithm will only need to maintain two types of significance maps. One is called the coefficient map F_C, which is used to store the locations of significant coefficients C. Another is called the descendant map F_D, which is used to store the locations of significant descendant sets D. During coding, if the LZC codec detects that coefficient $C(i,j)$ is significant against the threshold, the codec will mark $'1'$ on $F_C(i,j)$. Similarly, if the LZC codec finds one or more coefficients in descendant set $D(i,j)$ are significant, the codec will mark $'1'$ on $F_D(i,j)$. On the other hand, if $C(i,j)$ and $D(i,j)$ are insignificant, then $F_C(i,j)$ and $F_D(i,j)$ will be marked by $'0'$.

Figure 6.5 shows an example of how the significance maps work. In the wavelet matrix, blue blocks denote significant coefficients, so the same locations on the F_C map are marked by $'1'$; white blocks denote insignificant coefficients, so the same locations on the F_C map are marked $'0'$. Similarly, green blocks in Figure 6.5 denote a branch of a significant coefficient tree, so on the F_D map the location corresponding to the tree root is marked by $'1'$; on the other hand, gray blocks denote a branch of an insignificant coefficient tree, so on the F_D map the location corresponding to the tree root is marked by $'0'$.

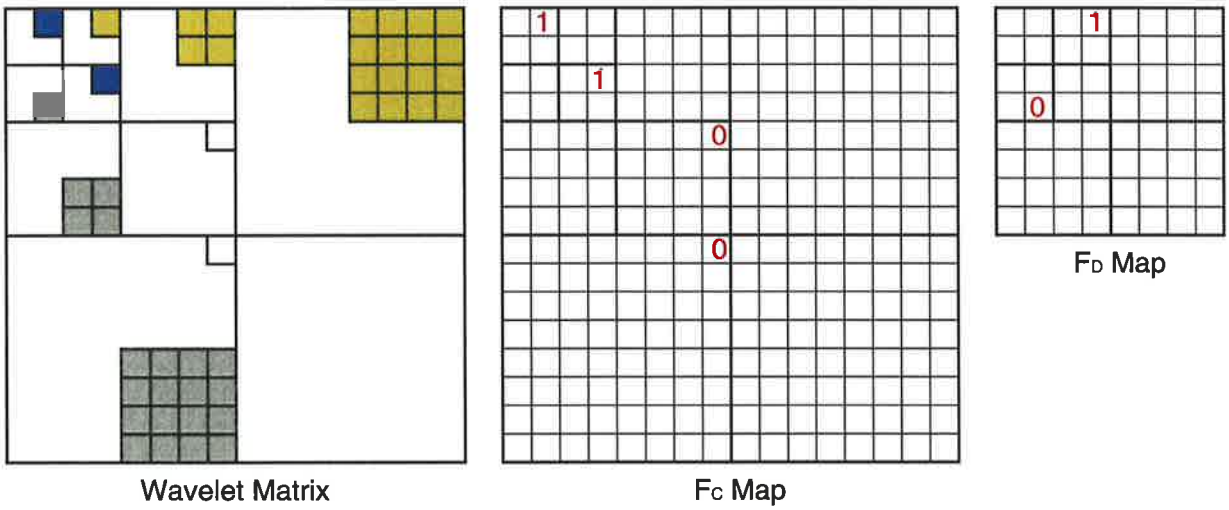


Figure 6.5: Significant Maps of LZC

From Figure 6.5 we can see that the F_C map has the same size as the wavelet matrix, while the F_D map is only a quarter of the wavelet matrix. This is because coefficients on the first transform level, *Level 1*, do not have any descendants. For coding color images, each color component will require its own F_C and F_D map. But even for a 512x512 color image, the total memory required is only 120KB [1], and it is fixed for all bit rates. On the other hand, as discussed in Section 5.5, SPIHT would require 3.375MB of memory for coding the same image. Therefore, LZC requires much less coding memory for storing the sorting information of wavelet coefficients.

6.4 LZC Algorithm Using Recursive Tree Search

The method used to search a wavelet coefficient tree has a direct impact on the bit-allocation efficiency, which in turn influences the reconstructed image quality. There are two search methods that can be used in the LZC algorithm. The first method is the *recursive* tree search, like the one used in SPIHT; the second method is the *raster* tree search, like the one used in EZW. The first version of the LZC algorithm exploits a recursive tree search, and will be discussed in this section. The second version of the algorithm exploits a raster tree search, and will be discussed in Section 6.5. The discussion in this section includes the recursive tree search order, the encoding procedures, and the decoding procedures.

[1] (512x512(F_c)+256x256(F_d)x3(colors))/8bits/1K = 120K

6.4.1 Recursive Tree Search Order

The advantage of using a recursive tree search is that the parent-child relationships of a wavelet coefficient tree are already embedded in the search order, so there is no need to store any information about the locations of tree coefficients. That is, when coding a wavelet matrix, the LZC codec will only need to know the coordinates of the root coefficients in the *LL* subband. From the coordinates of each root coefficient, the codec will be able to derive the coordinates for the rest of the tree coefficients by using the parent-child relationships mentioned in Section 6.3.1.

When the LZC codec performs the recursive tree search, it will search for the significant tree coefficients or tree sets in tree-depth fashion. The search always begins from tree roots, because they contain most visual information and are most likely to be significant. After that, the codec first searches the tree branches in the *HL* orientation followed by the branches in the *LH* and *HH* orientations. Also, within each tree branch, the search begins from the coarsest transform level and ends at the finest transform level. This search order is based on the discussion in Section 6.2.1. Figure 6.6 shows an example of the recursive tree search performed on a wavelet coefficient tree. The number marked on each coefficient indicates the order of visitation by the LZC codec. Therefore, the search order of the entire coefficient tree in this example follows the order

$$LL3 \rightarrow HL3 \rightarrow HL2 \rightarrow HL1 \rightarrow LH3 \rightarrow LH2 \rightarrow LH1 \rightarrow HH3 \rightarrow HH2 \rightarrow HH1, \text{ etc.}$$

To cover all of the wavelet coefficients in the search, the LZC codec performs a *semi*-raster search on the root coefficients in the *LL* subband. That is, the codec first visits a root coefficient, and from the root coefficient the codec starts to perform a recursive search on its coefficient tree (or descendant set). After finishing searching a tree, the codec simply moves to the next root coefficient in the raster position and performs the recursive tree search again, This process continues until the search has covered all of the wavelet coefficients.

Figure 6.7 gives an example of a search covering all of the wavelet coefficients. In this example, the wavelet matrix has four coefficient trees, which are indicated by four different colors. The LZC codec first visits the root of one color, and from that root the codec performs the recursive search on all the coefficients having the same color as the root. After the roots have been scanned once, all of the wavelet coefficients will also have been visited once by the LZC codec.

For coding color images, the same recursive search order is applied to each color component, but the LZC codec should follow the order of importance of each component as discussed in Section 6.2.2.

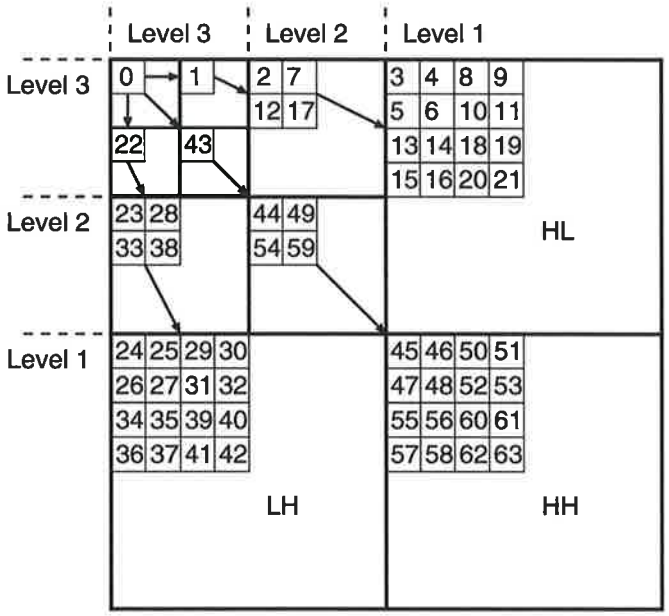


Figure 6.6: The recursive tree search order used in LZC algorithm

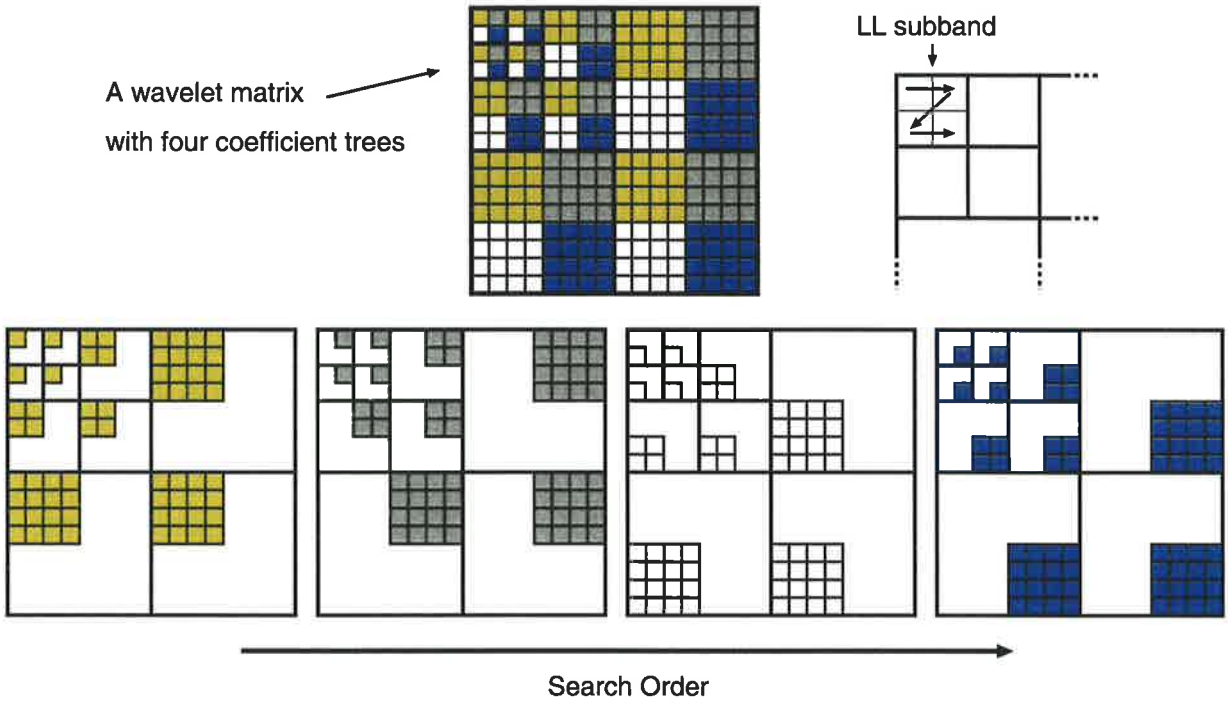


Figure 6.7: The search order of trees

6.4.2 Encoding Procedures

The encoding process of the LZC algorithm is separated into two procedures: the *main encoding procedure* and the *tree encoding procedure*. The function of the main encoding procedure is to control the coding flow and initiate the starting position for the recursive search on the coefficient trees. The function of the tree encoding procedure is to perform the recursive tree search on the coefficient trees by receiving the position coordinates from the main encoding procedure or from itself.

The LZC main encoding procedure for color image coding is presented in Algorithm 6.1. The color space used is YUV, but other color spaces can also be applied to this algorithm. Also, it should be noted that in the algorithm the significance test is performed by function $S_n()$, which is given by

$$S_n(T(i,j)) = \begin{cases} 1 & \text{if any } |C(i,j) \in T| \geq 2^n, \\ 0 & \text{otherweise,} \end{cases} \tag{6.5}$$

where $T(i,j)$ is either $C(i,j)$ or $D(i,j)$, and 2^n is the current coding threshold.

The main encoding procedure consists of four steps. In Step 1, the LZC codec clears all of the F_C and F_D maps for YUV components, and outputs a number n in bits, which is the word-length of the largest coefficient among the YUV components. The number n then gives the initial coding threshold as 2^n.

In Step 2, the LZC codec encodes the root coefficients in the *LL* subbands of the YUV wavelet matrixes. Suppose a root coefficient $C(i,j)$ is being coded. The LZC codec first checks the significance of $C(i,j)$ by looking at location (i,j) in the F_C map. If $F_C(i,j)$ is marked as $'1'$, then the root coefficient $C(i,j)$ has previously been found significant. Then the codec simply outputs a refinement bit from the nth bit of $C(i,j)$. On the other hand, if $F_C(i,j)$ is marked as $'0'$, the root coefficient $C(i,j)$ is not yet significant for any of the previous thresholds, so the LZC codec will test the significance of $C(i,j)$ with respect to the current threshold by using function $S_n()$. The result from $S_n()$ is written to the bit-stream. If the test result is insignificant, ie. $S_n(C(i,j)) = 0$, the codec will simply finish the encoding on coefficient $C(i,j)$, and move to the next root coefficient. On the other hand, if the test result is significant, ie. $S_n(C(i,j)) = 1$, the codec will output the sign of $C(i,j)$ and mark location (i,j) in the F_C map to $'1'$, ie. $F_C(i,j) = 1$, for magnitude refinement. The encoding in Step 2 finishes when all of the root coefficients of the YUV components have been scanned once.

In Step 3, the LZC codec performs encoding on the wavelet coefficient trees, and the descendant sets of the root coefficients. Suppose a root coefficient is $C(i,j)$, then its descendant set will be represented as $D(i,j)$. The LZC codec first needs to check the significance of $D(i,j)$

by looking at location (i, j) in the F_D map. If $F_D(i, j)$ is marked as '1', then one or more coefficients in the descendant set $D(i, j)$ have been found significant for the previous thresholds. Therefore, the codec partitions the tree by passing the coordinates of children coefficients in $O(i, j)$ to the tree encoding procedure. On the other hand, if $F_D(i, j)$ is marked as '0', that means all of the coefficients in $D(i, j)$ are not yet significant. Therefore, the codec tests the significance of $D(i, j)$ by using function $S_n()$, and the test result is written to the bit-stream. If the test is insignificant, ie. $S_n(D(i, j)) = 0$, the codec will finish the encoding on $D(i, j)$ and move to the next tree. However, if the test is significant, ie. $S_n(D(i, j)) = 1$, then the codec will mark '1' in $F_D(i, j)$ and partition the tree by passing the coordinates of children coefficients in $O(i, j)$ to the tree encoding procedure. Step 3 finishes when all of the wavelet coefficient trees of the YUV matrixes have been searched once.

In Step 4, the LZC codec decreases the number n by 1, which halves the coding threshold. After that, the codec repeats all of the encoding process from Step 2, again.

The tree encoding procedure of the LZC algorithm is shown in Algorithm 6.2. It performs two tasks: encoding an individual coefficient and encoding the tree branches by partition. The process for encoding an individual coefficient is similar to Step 2 in the main encoding procedure, and the process for encoding the tree branches is similar to Step 3 in the main encoding process. However, the difference for the tree branch encoding process in the tree encoding procedure is that the partition on tree branches is performed by recursively calling the procedure itself. The recursive procedure calls continue until all of the tree branches have been partitioned into individual coefficients. That is, recursion will stop at *level 2*, as coefficients in *level 1* do not have any descendants.

In Figure 6.8 the tree encoding procedure is represented diagrammatically. Suppose a tree branch is being encoded, the LZC codec performs corresponding encoding processes based on the significance marked in the F_C or F_D map. The bits written to the bit-stream are shown as '0' and '1', and are colored in red. The codec first performs encoding on the coefficient $C(i, j)$, followed by its tree branch $D(i, j)$. The partition of the tree branch $D(i, j)$ continues until all of the tree branches have been partitioned into individual coefficients, such as shown in the diagram.

The progressiveness of the reconstructed image quality in LZC algorithm is achieved by performing bit-layer coding. As already mentioned, the number n gives the initial threshold as 2^n, as well as the number of bit-layers for YUV wavelet matrixes. In the first coding run, the LZC codec will encode those coefficients having their MSB bits in the nth bit-layer, ie. those having their magnitude in the range of $2^{n+1} - 1$ to 2^n. After that, the LZC codec decreases the number n by one, so the threshold becomes 2^{n-1}. Then, the LZC codec will encode those coefficients having their MSB bits in the $(n - 1)$th bit-layer, ie. those having their magnitude

```
Procedure MainEncoding
Step 1:
  for all Y, U, and V components
   clear F_C and F_D;
   output n = ⌊log2(max_(i,j)|C(i,j)|)⌋;
Step 2:
  for all Y, U, and V components
   for all C(i,j) ∈ R do
     if F_C(i,j) = 1 then
       output the nth significant bit of |C(i,j)|;
     else
       output S_n(C(i,j));
       if S_n(C(i,j)) = 1 then
         output the sign of C(i,j)
         and set F_C(i,j) = 1;
Step 3:
  for all Y, U, and V components
   for all D(i,j) ∈ R do
     if F_D(i,j) = 1 then
       for each C(k,l) ∈ O(i,j)
       → EncodeTree(k,l,n);
     else
       output S_n(D(i,j));
       if S_n(D(i,j)) = 1 then
         set F_D(i,j) = 1;
           for each C(k,l) ∈ O(i,j)
           → EncodeTree(k,l,n);
Step 4:
  decrease n by 1 and go back to Step 2;
End
```

Algorithm 6.1: LZC Main Encoding Procedure

```
Procedure EncodeTree(i,j,n)
 if F_C(i,j) = 1 then
    output the nth significant bit of |C(i,j)|;
 else
    output S_n(C(i,j));
    if S_n(C(i,j)) = 1 then
       output the sign of C(i,j) and
       set F_C(i,j) = 1;

 if (i,j) NOT IN Level 1
    if F_D(i,j) = 1 then
       for each C(k,l) ∈ O(i,j)
        →EncodeTree(k,l,n);
    else
       output S_n(D(i,j));
       if S_n(D(i,j)) = 1 then
          set F_D(i,j) = 1;
          for each C(k,l) ∈ O(i,j)
          →EncodeTree(k,l,n);
End
```

Algorithm 6.2: LZC Tree Encoding Procedure

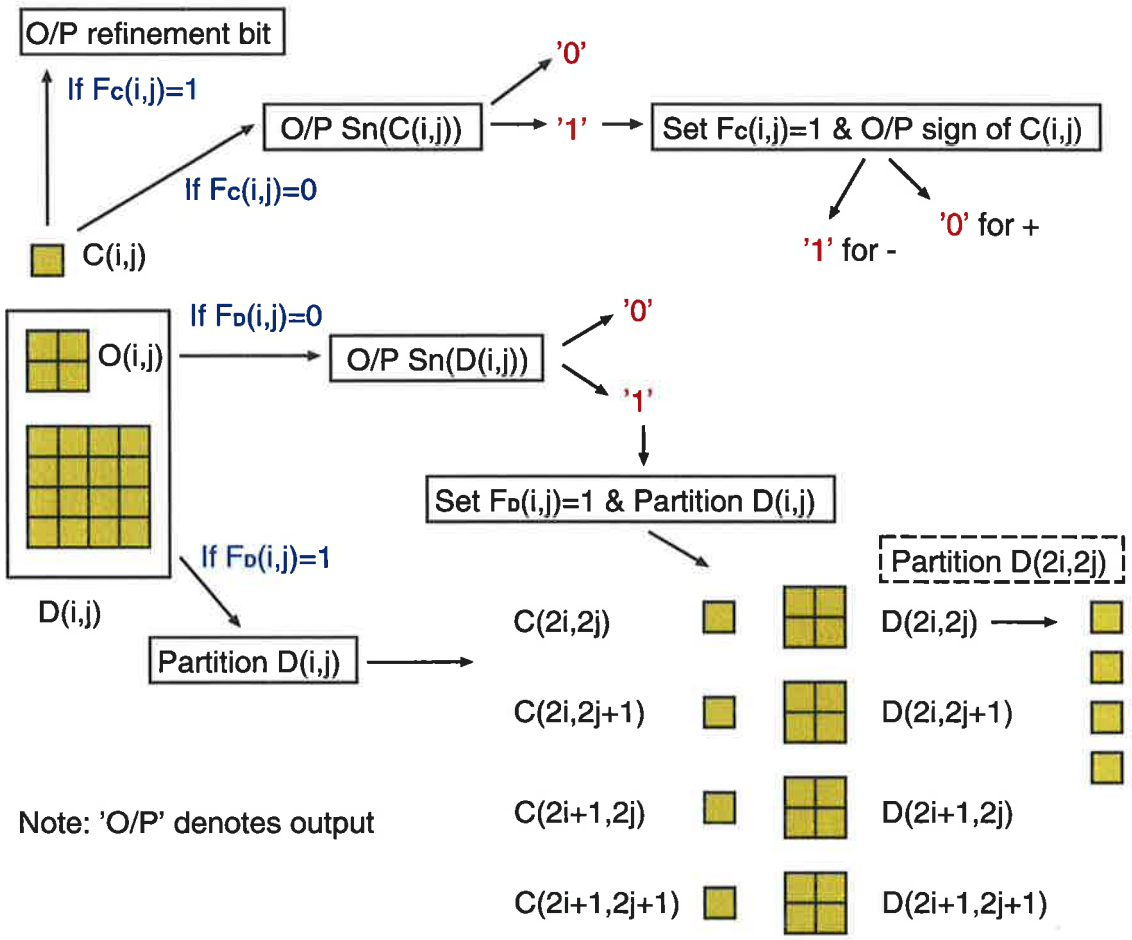


Figure 6.8: LZC tree encoding procedure

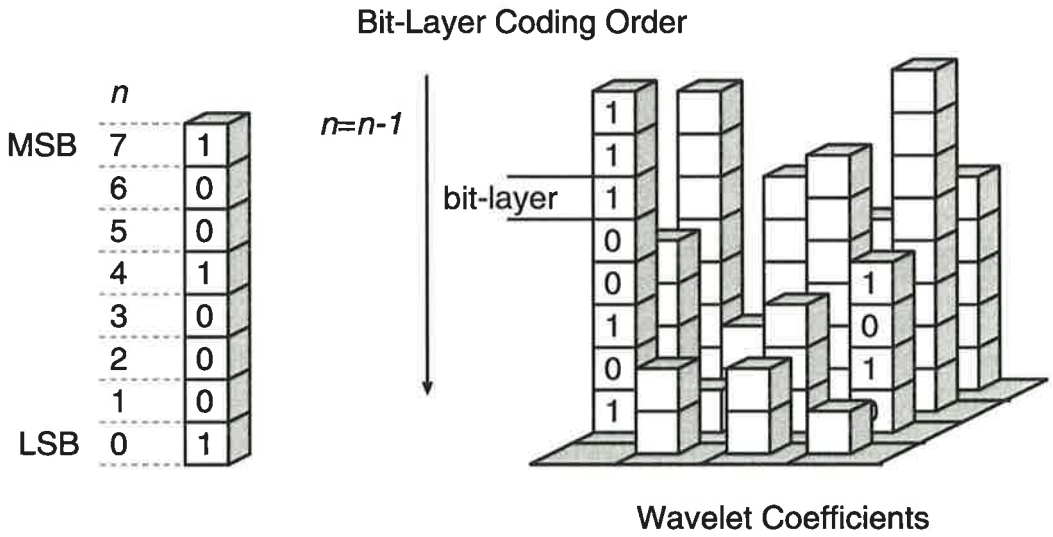


Figure 6.9: LZC bit-layer coding order

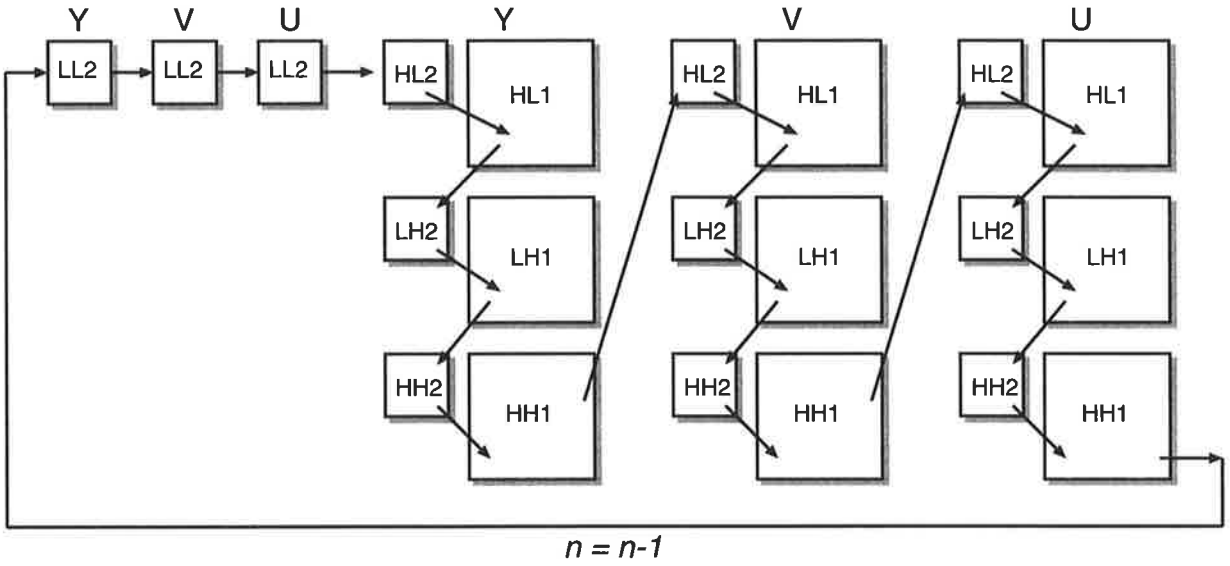


Figure 6.10: Example of LZC encoding order on YUV wavelet matrixes

in the range of $2^n - 1$ to 2^{n-1}. Besides encoding new significant coefficients, the LZC codec will also output the $(n-1)$th bits for those having been found significant previously. The LZC codec repeats this bit-layer coding process until it runs out of coding bit-budget or reaches the 0th bit-layer, ie. the lowest bit-layer. One example of the bit-layer coding order is illustrated in Figure 6.9.

The LZC encoding order on YUV wavelet matrices begins on Y, followed by V and then U, and within the wavelet matrix of each color components the encoding begins in the *LL* subband, followed by the *HL*, *LH*, and *HH* subbands. As discussed in Section 6.2, this color encoding order follows the characteristics of our visual system. Figure 6.10 illustrates the LZC color encoding order in a block form. After all of the matrices have been scanned once, the LZC codec will move to the next bit-layer and repeat the same encoding order. Therefore, after encoding several bit-layers, the bit-stream generated by the LZC codec will look like the one shown in Figure 6.11. Since, the LZC color bit-stream has the information ordering property, ie. embeddedness, it can be truncated at any point during decoding.

6.4.3 Decoding Procedures

The pseudo code of the LZC *main decoding procedure* and *tree decoding procedure* are shown in Algorithm 6.3 and Algorithm 6.4, respectively. Both decoding procedures look very similar to the corresponding encoding procedures. The only two differences are that the *output* process in encoding is changed to an *input* process in decoding, and the coefficient significance test

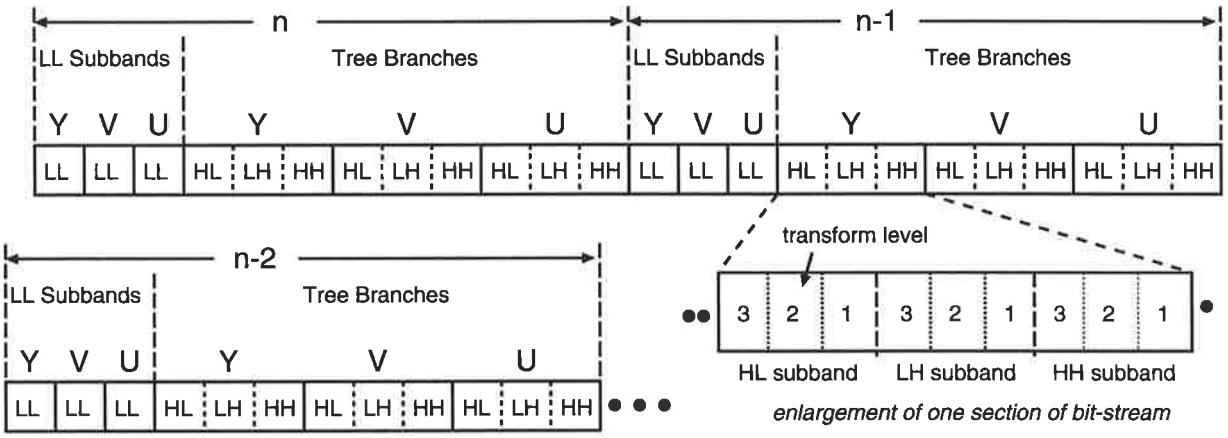


Figure 6.11: LZC color bit-stream after encoding several bit-layers

in encoding is changed to coefficient value reconstruction in decoding. Other than these two differences, the coding orders for both encoding and the decoding are identical. Consequently, the LZC algorithm has the same encoding and decoding complexity.

Since the decoding process is the same as the encoding one, here we only discuss how to reconstruct and refine a wavelet coefficient. Suppose a coefficient $C(i,j)$ is not yet significant, ie. $F_C(i,j)$ is still marked as '0'. If the LZC codec inputs a bit from the bit-stream representing a *significant* test result, ie. $S_n(C(i,j)) = 1$, then the codec will input an additional bit representing the *sign* bit and reconstruct $C(i,j)$ as

$$C(i,j) = (sign)\ 1.5 \times 2^n, \tag{6.6}$$

where $sign$ is $+$ if the sign bit is equal to '0', or $-$ if the sign bit is equal to '1' , as indicated in Figure 6.8. After that, the codec will mark '1' on $F_C(i,j)$.

On the other hand, if coefficient $C(i,j)$ is already significant, ie. $F_C(i,j) = 1$, the LZC codec will then input the refinement bit from the bit-stream and refine $C(i,j)$. The magnitude refinement for $C(i,j)$ is given by

$$C(i,j) = \begin{cases} (sign)\ (|C(i,j)| - 2^{n-1}) & \text{if refinement bit} =' 0', \\ (sign)\ (|C(i,j)| + 2^{n-1}) & \text{if refinement bit} =' 1', \end{cases} \tag{6.7}$$

where $(sign)$ is the sign of $C(i,j)$, and n denotes that the decoding is at the nth bit-layer. We should note that the maximum *quantization error* for each coefficient is 2^{n-1} after magnitude refinement, and will be halved when moving one bit-layer down.

```
Procedure MainDecoding
Step 1:
  input n;
  clear F_C and F_D;
Step 2:
  for all Y, U, and V components
   for all C(i,j) ∈ R do
     if F_C(i,j) = 1 then
       input the nth significant bit of |C(i,j)| and
       refine C(i,j);
     else
       input S_n(C(i,j));
       if S_n(C(i,j)) = 1 then
         input the sign of C(i,j) and
         reconstruct C(i,j) = (sign) 1.5 × 2^n;
         set F_C(i,j) = 1;
Step 3:
  for all Y, U, and V components
   for all D(i,j) ∈ R(i,j) do
     if F_D(i,j) = 1 then
       for each C(k,l) ∈ O(i,j)
       → DecodeTree(k,l,n);
     else
       input S_n(D(i,j));
       if S_n(D(i,j)) = 1 then
         set F_D(i,j) = 1;
           for each C(k,l) ∈ O(i,j)
           → DecodeTree(k,l,n);
Step 4:
  decrease n by 1 and go back to Step 2;
End
```

Algorithm 6.3: LZC Main Decoding Procedure

```
Procedure DecodeTree(i,j,n)
 if F_C(i,j) = 1 then
    input the nth significant bit of |C(i,j)|
    and refine C(i,j);
 else
    input S_n(C(i,j));
    if S_n(C(i,j)) = 1 then
       input the sign of C(i,j) and
       reconstruct C(i,j) = (sign) 1.5 × 2^n;
       set F_C(i,j) = 1;

 if (i,j) NOT IN Level 1
    if F_D(i,j) = 1 then
       for each C(k,l) ∈ O(i,j)
        →DecodeTree(k,l,n);
    else
       input S_n(D(i,j));
       if S_n(D(i,j)) = 1 then
          set F_D(i,j) = 1;
          for each C(k,l) ∈ O(i,j)
          →DecodeTree(k,l,n);
End
```

Algorithm 6.4: LZC Tree Decoding Procedure

6.5 LZC Algorithm Using Raster Tree Search

This section will discuss the second version of the LZC algorithm, which exploits a raster tree search in the coding process. The discussion in this section includes the raster tree search used in the LZC algorithm, the benefits of using the raster tree search, and how the LZC algorithm should be modified in order to perform a raster tree search.

6.5.1 Raster Tree Search Order

The raster tree search used in the second version of the LZC algorithm begins from the coarsest transform level to the finest transform level. Within each transform level, the orientation search order begins from the *HL* subband, followed by the *LH* and *HH* subbands. One example of the raster tree search on a 3-level transformed wavelet matrix is shown in Figure 6.12. For this example, the subband search order is

$$LL3 \rightarrow HL3 \rightarrow LH3 \rightarrow HH3 \rightarrow HL2 \rightarrow LH2 \rightarrow HH2 \rightarrow HL1 \rightarrow LH1 \rightarrow HH1, \text{ etc.}$$

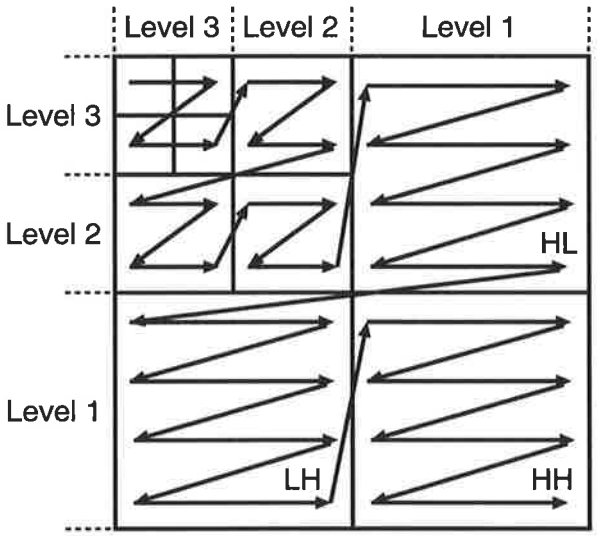


Figure 6.12: Raster tree search order of LZC

There are two reasons why the raster tree search is more appropriate than the recursive tree search for the bit-allocation scheme. First, as discussed in Section 6.2.1, our visual system is more sensitive to low spatial frequency signals, and also to horizontal and vertical signals. Therefore, the raster search order follows exactly the order of wavelet coefficients that descend in visual importance. As a result, by using a raster tree search, the LZC codec will be able to allocate more bit-budget to encode the visually more important wavelet coefficients. In contrast,

a recursive tree search can only search the wavelet tree in one orientation before another. Therefore, to use a recursive tree search, the LZC codec may waste the bit-budget encoding visually unimportant wavelet coefficients in the higher frequency subbands of one orientation, instead of encoding visually more important wavelet coefficients in the lower frequency subbands of another orientation.

Second, after the wavelet transform, coefficients in the lower frequency subbands normally have larger magnitudes than those in the higher frequency subband. Since coefficients with larger magnitudes are more significant, they should be encoded first for maintaining the information ordering property of the bit-stream, ie. maintaining the embeddedness. Therefore, by using the raster tree search, the LZC codec will be able to generate a better embedded bit-stream, as the search follows the order that wavelet coefficients descend in magnitude. On the other hand, the recursive tree search does not follow the order of significance, so the bit-allocation scheme may not generate a good embedded bit-stream.

For those two reasons, the raster tree search is better than the recursive tree search and should produce more uniform quality on the reconstructed images. This is because for the raster tree search, coefficients with the same visual importance and the same magnitude significance are encoded at the same time, whereas, for the recursive tree search, only coefficients with the same spatial location, rather than the same importance, are encoded at the same time.

Figure 6.13 illustrates how the reconstructed visual quality can be affected by the recursive and raster tree search methods. If the LZC codec uses the recursive tree search in its coding process and stops coding at the point indicated, then the upper spatial locations of all subbands, as colored in white, will have been coded one bit-layer lower than the lower spatial locations, as colored in light-blue. Therefore, wavelet coefficients in the white regions will have a finer magnitude precision than those in the light-blue regions. Consequently, the reconstructed image will have a better visual quality in the spatial location corresponding to the white regions than in that corresponding to the light-blue regions. In particular at the low bit-rate when the coding process is only performed on few bit-layers, the uneven reconstructed visual quality will be very noticeable. This is because the precision difference between each bit-layer is given by 2^{n-1}, and the larger the value of n the larger the precision difference.

On the other hand, if the LZC codec uses the raster tree search and stops coding at the point indicated, then all of the lower spatial frequency subbands are one bit-layer lower than the higher frequency subbands. This means that wavelet coefficients of the same importance are coded at the same magnitude precision, so the visual quality on the reconstructed image will look more uniform. Despite there being a precision difference in one high frequency subband, because our visual system has a low contrast sensitivity towards high spatial frequency signals, the difference on the reconstructed visual quality will be less noticeable. Hence, the raster tree

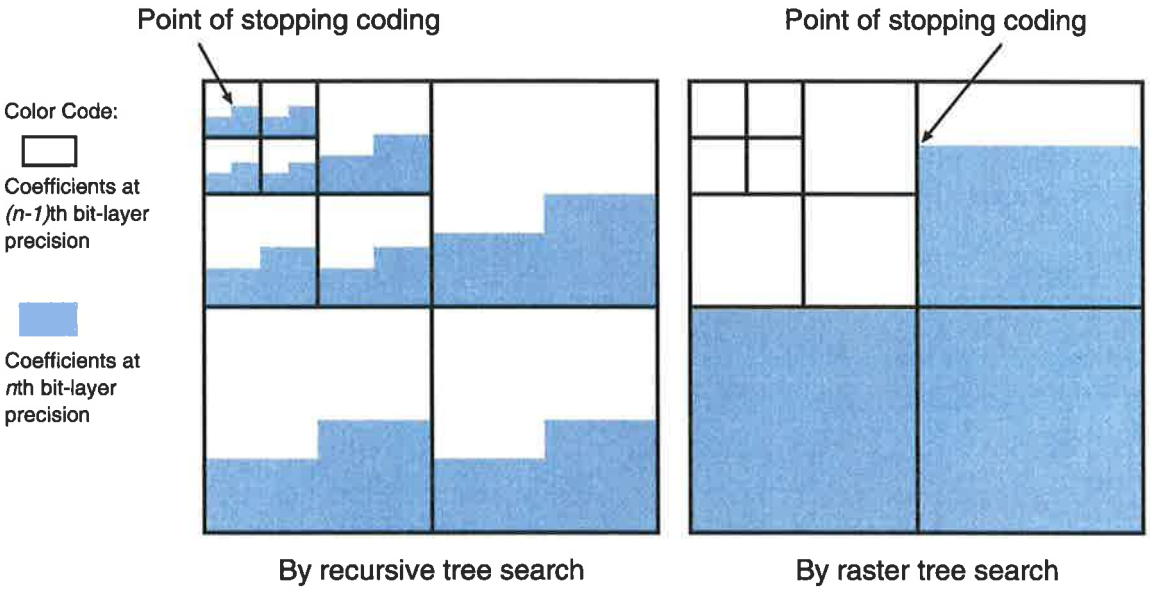


Figure 6.13: Bit-layer coding for the raster and the recursive tree search

search is more suitable than the recursive tree search for producing an image with more uniform visual quality at low coding bit-rates.

6.5.2 The Tree-Pruning Process

The raster tree search is more appropriate for bit-allocation process for low bit-rate coding, but the zerotree structure is no longer embedded in the search order. Therefore, there is a need to have an additional testing process to determine the zerotree relations and some additional buffer memory to store the test results. In order to minimize the increase of coding complexity associated with the additional testing process, the LZC algorithm uses a simple *tree-pruning* process to construct the zerotree relations. Also, for maintaining minimal memory overheads, the LZC algorithm uses an additional bit-map to store the zerotree relations. This bit-map is called the F_T map, where T stands for *tree*. For clarity, the second version of the LZC algorithm is called Tree-Pruning Listless Zerotree Coding (TPLZC).

The working mechanisms of the tree-pruning process and the function of the F_T map are explained as follows. Initially the F_T map does not contain any coefficient tree structures, as each location in the map is set to $'0'$. Therefore, at the beginning, the TPLZC codec actually needs to perform a *tree-growing* process. The tree-growing process is used to construct the significant trees that are rooted in the LL subband. If a tree (or descendant set) $D(i,j)$ is significant, ie. $S_n(D(i,j)) = 1$, the TPLZC codec will grow a significant tree in the F_T map by marking those locations corresponding to $D(i,j)$ to $'1'$. However, if $D(i,j)$ is insignificant, ie.

$S_n(D(i,j)) = 0$, the codec simply ignores this tree and moves to next one. After growing all of the significant trees in the F_T map, the TPLZC can then decide to keep or prune away the tree branches by their significance along the raster search order.

The tree-pruning process is performed when the TPLZC codec searches the branches of significant trees. Consider when a tree branch $D(i,j)$ is coded. If $D(i,j)$ is significant, the codec will then keep it, so that the TPLZC codec can keep searching on $D(i,j)$ for significant coefficients or tree branches. On the other hand, if $D(i,j)$ is insignificant, then the codec will code it as a zerotree and will prune it away. Therefore, in the F_T map those locations that correspond to $D(i,j)$ will be marked as '0', so that the TPLZC codec can bypass those locations. This is because the codec already knows that no significant coefficients or tree branches can be found in $D(i,j)$.

Figure 6.14 gives an example of the tree-growing and tree-pruning process. Figure 6.14(a) shows that the first coefficient tree, which is colored in green, is significant. Then, the TPLZC codec first marks the tree root in the F_D map to '1', denoted by the light-blue color, and grows this significant tree in the F_T map by marking '1' on the corresponding tree locations. Figure 6.14(b) shows the results of the tree-growing process in the *LL* subband. Except for the last tree, represented by white blocks, the rest of the trees have been found significant and have been grown in the F_T map. The root locations corresponding to the significant tree in the F_D map are also marked by '1'. Figure 6.14(c) shows the results after the TPLZC codec has pruned away some insignificant tree branches by marking their locations in the F_T map to '0', as denoted by the white color. Those insignificant tree branches are indicated on the wavelet matrix. The figure also demonstrates that the TPLZC codec only performs a raster search on the coefficients with their locations in the F_T map indicated by '1'. Those with their locations in the F_T map marked by '0' have been bypassed by the codec, as illustrated by the red arrows.

Since the results from the significant test, $S_n()$, are marked in the F_D map, the only complexity overhead for TPLZC algorithm to achieve the raster tree search is marking '1' or '0' in the F_T map. Furthermore, for both gray-scale and color images, the additional coding memory for the F_T map is only $N/8$ Bytes, where N is the total number of image pixels. Moreover, the procedure to mark zerotree regions in the F_T map is performed soon after the tree significance test so the number of coding passes is still kept to one. Hence, the TPLZC requires only little additional coding memory and minimal coding complexity to achieve the raster tree search.

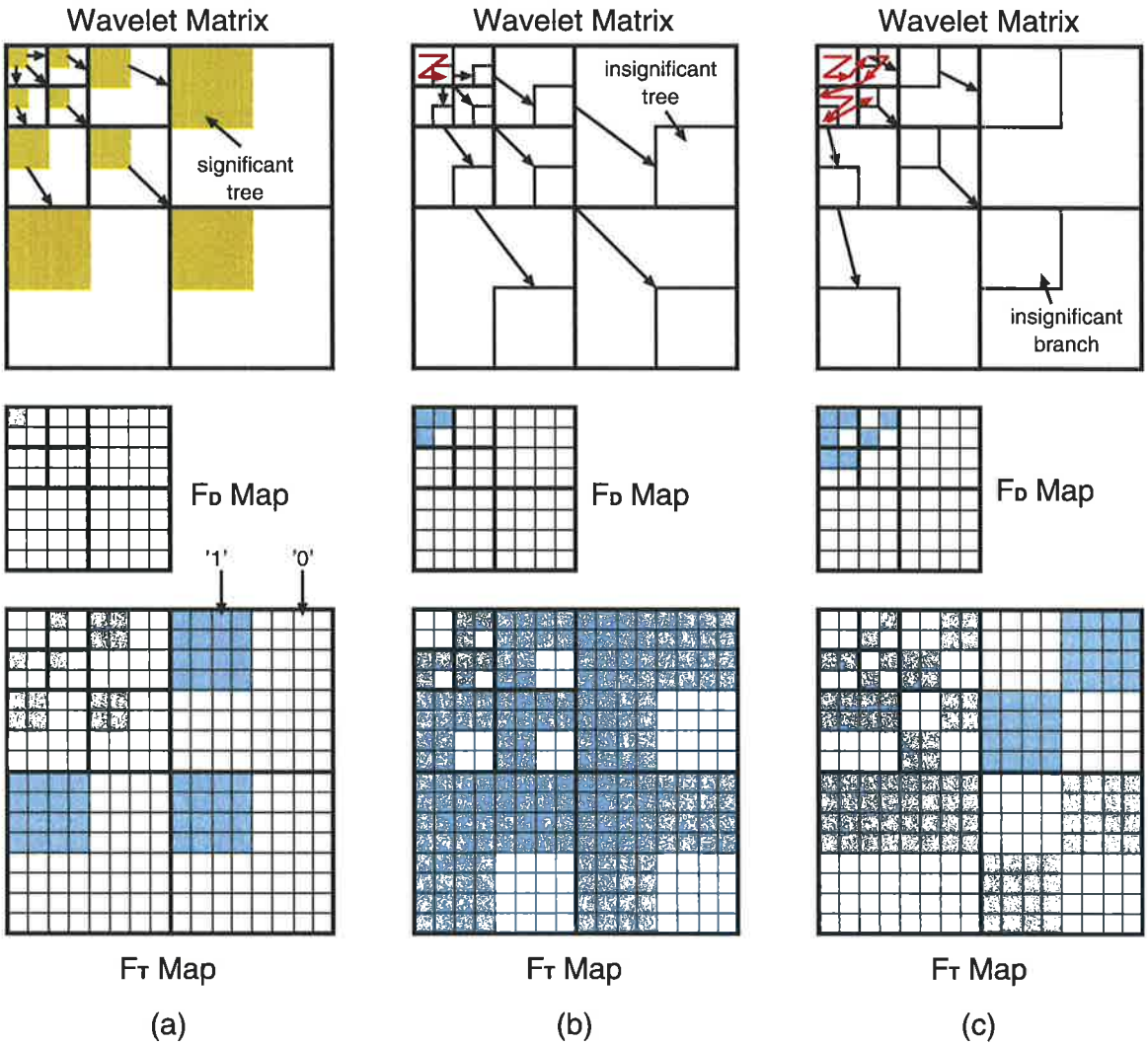


Figure 6.14: The Tree-growing and tree-pruning process (a) growing the first significant tree on the F_T map (b) finishing the tree-growing process in the *LL* subband (c) keeping or pruning away the tree branches

6.5.3 Encoding Procedures

The TPLZC *main encoding procedure* for color image coding is presented in Algorithm 6.5. As in LZC, the TPLZC main encoding procedure also consists of four steps.

In Step 1 and Step 2, the TPLZC codec performs exactly the same tasks as the LZC codec, so we just briefly mention those tasks here. In Step 1 the TPLZC codec outputs the number n for determining the initial threshold and clears the F_C and F_D maps for each color component. In Step 2, the TPLZC performs the encoding on the root coefficients, including detecting and encoding significant root coefficients, and refining the magnitude of significant coefficients.

In Step 3, since the wavelet coefficient trees are coded in raster order, the tasks performed by the TPLZC codec are slightly different from those performed by the LZC codec. First, before coding each color component, the TPLZC codec first clears the F_T map for reconstructing the tree structures for that color component. Suppose a coefficient tree $D(i,j)$ is encoded, the TPLZC codec checks its significance in the F_D map. If $D(i,j)$ has been found significant in the previous coding run, ie. $F_D(i,j) = 1$, then the TPLZC codec will grow the tree by setting the corresponding locations in the F_T map to '1'. On the other hand, if $D(i,j)$ has not yet been found significant, then the TPLZC codec will test its significance by function $S_n()$. If $D(i,j)$ is significant for the current threshold, then the TPLZC codec will mark its root in the F_D map to '1' and grow the significant tree in the F_T map. If $D(i,j)$ is insignificant, the TPLZC codec simply ignores that tree and moves on to test the next one. After all of the significant trees have been grown, the TPLZC codec will perform the tree branch encoding by calling the *tree encoding procedure*. The tree branch encoding is performed subband by subband, following the raster order, ie. from *HL* to *LH* to *HH*, and from the highest to the lowest transform level. The encoding in this step ends when all three color components have been searched and encoded.

In Step 4, the TPLZC codec decreases the number n by 1 and repeats the encoding from Step 2 again for coding the next bit-layer.

The TPLZC tree encoding procedure is presented in Algorithm 6.6. It is mainly responsible for two tasks: encoding individual coefficients in a subband and pruning away insignificant tree branches rooted in that subband. At the beginning, the TPLZC codec uses function $Subband(L,S)$ to obtain all of the coefficient coordinates in that subband. In function $Subband(L,S)$, L denotes the wavelet transform level, and S denotes the subband orientation like HL, LH or HH. Figure 6.15 gives an example of the denotation of function $Subband(L,S)$.

Suppose coefficient $C(i,j)$ is encoded, then the TPLZC codec first checks the coefficient status in the F_T map. If $C(i,j)$ descends from a zerotree root, ie. $F_T(i,j) = 0$, then the TPLZC codec will skip the encoding of $C(i,j)$ and its descendant set $D(i,j)$. On the other hand, if $C(i,j)$ is a member of a significant tree, ie. $F_T(i,j) = 1$, the TPLZC codec will then proceed to encode coefficient $C(i,j)$ and its descendant set $D(i,j)$. For the encoding of $C(i,j)$, the codec either refines its magnitude or detects its significance. For the encoding of $D(i,j)$ the codec either keeps it or prunes it away, depending on its significance.

The difference between the LZC and TPLZC tree encoding procedures is that the LZC tree encoding procedure is called by the main encoding procedure and recursively called by itself, whereas the TPLZC tree encoding procedure is only called by the main encoding procedure. Another difference is that the LZC tree encoding procedure only deals with one tree branch; while, the TPLZC tree encoding procedure deals with a group of tree branches which are rooted

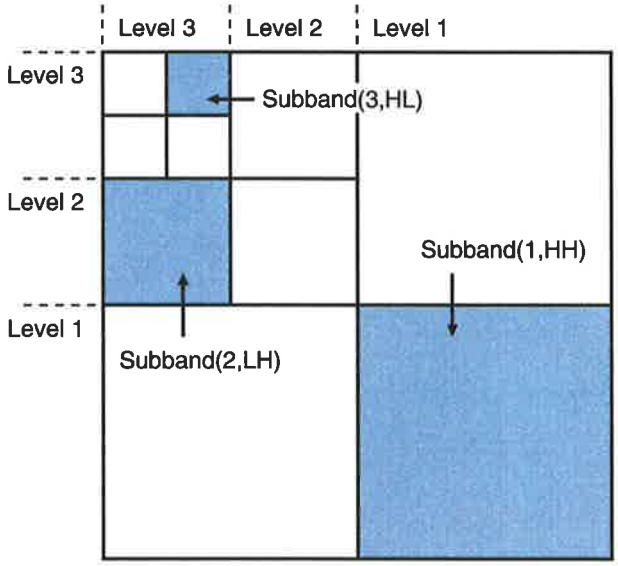


Figure 6.15: Examples of the denotation of function $Subband(L, S)$

in the same subband.

Similarly to LZC, the TPLZC encoding order on YUV wavelet matrixes also begins from Y, followed by V and then U. However, the encoding within each wavelet matrix is in raster order, so the color encoding order performed by the TPLZC codec is like the block diagram shown in Figure 6.16. Therefore, after encoding several bit-layers, the bit-stream generated by the TPLZC codec will look like the one shown in Figure 6.17. The TPLZC bit-stream has a better information ordering property than the LZC bit-stream, because it follows the exact order in which the information decreases in visual importance and magnitude significance.

6.5.4 Decoding Procedures

Since the TPLZC decoding process follows exactly same order as the encoding process and also the coefficient reconstruction is the same as the LZC decoding process, we will just present the TPLZC decoding procedures is this section without further discussion.

The TPLZC main decoding procedure and tree decoding procedure are presented in Algorithm 6.7 and Algorithm 6.8, respectively.

```
Procedure MainEncoding
Step 1:
  for all Y, U, and V components
   clear F_C and F_D;
   output n = ⌊log2(max_(i,j)|C(i,j)|)⌋;
Step 2:
  for all Y, U, and V components
   for all C(i,j) ∈ R do
     if F_C(i,j) = 1 then
       output the nth significant bit of |C(i,j)|;
     else
       output S_n(C(i,j));
       if S_n(C(i,j)) = 1 then
         output the sign of C(i,j)
         and set F_C(i,j) = 1;
Step 3:
  for all Y, U, and V components
   clear F_T;
   for all D(i,j) ∈ R do
     if F_D(i,j) = 1 then
       for each C(k,l) ∈ D(i,j)
       → set F_T(k,l) = 1;  // tree-growing process
     else
       output S_n(D(i,j));
       if S_n(D(i,j)) = 1 then
         set F_D(i,j) = 1;
           for each C(k,l) ∈ D(i,j)
           → set F_T(k,l) = 1;  // tree-growing process
   for transform level L = highest to 1
     for subband S = HL, LH and HH
       → EncodeTree(L,S,n);
Step 4:
  decrease n by 1 and go back to Step 2;
End
```

Algorithm 6.5: TPLZC Main Encoding Procedure

```
Procedure EncodeTree(L,S,n)
 for all (i,j) ∈ Subband(L,S)
   if F_T(i,j) = 1
     if F_C(i,j) = 1 then
        output the nth significant bit of |C(i,j)|;
     else
        output S_n(C(i,j));
        if S_n(C(i,j)) = 1 then
           output the sign of C(i,j) and
           set F_C(i,j) = 1;

     if L ≠ 1 and F_D(i,j) = 0
        output S_n(D(i,j));
        if S_n(D(i,j)) = 1 then
           set F_D(i,j) = 1;
        else
           for each C(k,l) ∈ D(i,j)
           → set F_T(k,l) = 0;  // tree-pruning process
End
```

Algorithm 6.6: TPLZC Tree Encoding Procedure

```
Procedure MainDecoding
Step 1:
  input n;
  clear F_C and F_D;
Step 2:
  for all Y, U, and V components
   for all C(i,j) ∈ R do
     if F_C(i,j) = 1 then
       input the nth significant bit of |C(i,j)|
       and refine C(i,j);
     else
       input S_n(C(i,j));
       if S_n(C(i,j)) = 1 then
         input the sign of C(i,j) and
         reconstruct C(i,j) = (sign) 1.5 × 2^n;
         and set F_C(i,j) = 1;
Step 3:
  for all Y, U, and V components
   clear F_T;
   for all D(i,j) ∈ R(i,j) do
     if F_D(i,j) = 1 then
       for each C(k,l) ∈ D(i,j)
       → set F_T(k,l) = 1;  // tree-growing process
     else
       input S_n(D(i,j));
       if S_n(D(i,j)) = 1 then
         set F_D(i,j) = 1;
           for each C(k,l) ∈ D(i,j)
           → set F_T(k,l) = 1;  // tree-growing process
   for transform level L = highest to 1
     for subband S = HL, LH and HH
       → DecodeTree(L,S,n);
Step 4:
  decrease n by 1 and go back to Step 2;
End
```

Algorithm 6.7: TPLZC Main Decoding Procedure

```
Procedure DecodeTree(L,S,n)
 for all (i,j) ∈ Subband(L,S)
   if F_T(i,j) = 1
     if F_C(i,j) = 1 then
        input the nth significant bit of |C(i,j)|
        and refine C(i,j);
     else
        input S_n(C(i,j));
        if S_n(C(i,j)) = 1 then
           input the sign of C(i,j) and
           reconstruct C(i,j) = (sign) 1.5 × 2^n;
           set F_C(i,j) = 1;

     if L ≠ 1 and F_D(i,j) = 0
        input S_n(D(i,j));
        if S_n(D(i,j)) = 1 then
           set F_D(i,j) = 1;
        else
           for each C(k,l) ∈ D(i,j)
           → set F_T(k,l) = 0;  // tree-pruning process
End
```

Algorithm 6.8: TPLZC Tree Decoding Procedure

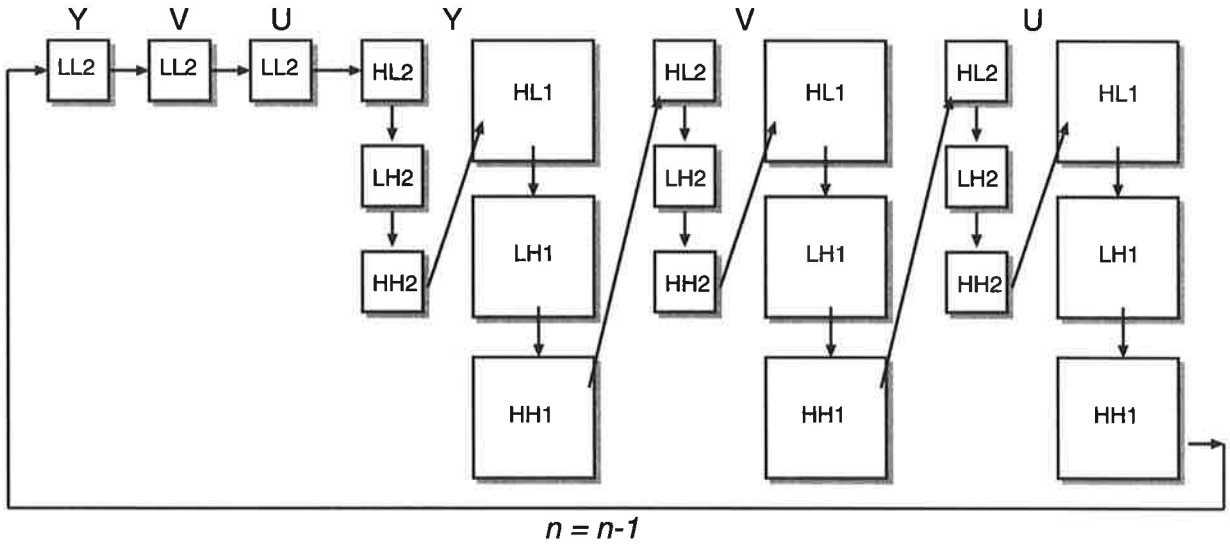


Figure 6.16: Example of TPLZC encoding order on YUV wavelet matrixes

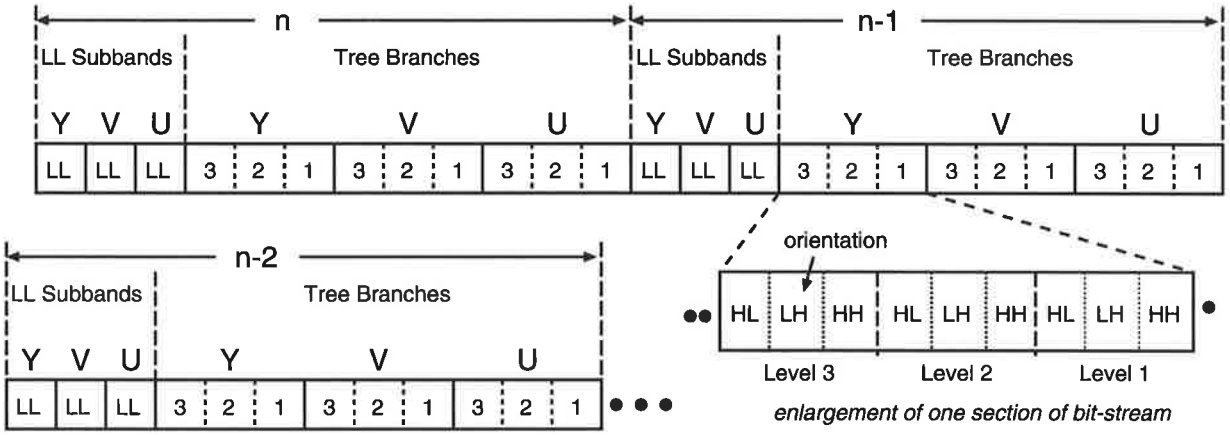


Figure 6.17: TPLZC color bit-stream after encoding several bit-layers

6.5.5 An Alternative Tree-Pruning Process

As discussed in Section 6.5.2, initially every location in the F_T map is set to '0', so the TPLZC codec needs to grow significant trees in the F_T map first before commencing the coding.

Alternatively, we can set every location in the F_T map to '1', by assuming that all of the coefficient trees rooted on the *LL* subband are significant, and after knowing their significance, the TPLZC codec can then prune away the insignificant trees. To implement this alternative tree-pruning process, we can substitute the code segment shown in Algorithm 6.9 for Step 3 in the TPLZC main encoding procedure.

For this alternative implementation, the TPLZC codec first *sets* every location in the F_T map

to '1', by assuming every tree is significant. After that the codec checks the tree's significance in the F_D map. If in the F_D map the tree has been coded as a significant tree, then our assumption is correct, so the codec will keep the tree. Otherwise, the codec will test the tree's significance. If the tree is significant now, it will be kept and the codec will set its root in the F_D map to '1'. However, if the tree is still insignificant, then the codec will prune this tree away by marking the corresponding locations in the F_T map to '0'.

For decoding, we can also replace Step 3 in the TPLZC main decoding procedure with the one in Algorithm 6.9, and just change 'EncodeTree' to 'DecodeTree' in the last line.

```
Step 3:
  for all Y, U, and V components
   set F_T;
   for all D(i,j) ∈ R do
     if F_D(i,j) = 0 then
       output S_n(D(i,j));
       if S_n(D(i,j)) = 1 then
         set F_D(i,j) = 1;
       else
         for each C(k,l) ∈ D(i,j)
          → set F_T(k,l) = 0;  // tree-pruning process
   for transform level L = highest to 1
     for subband S = HL, LH and HH
       → EncodeTree(L,S,n);
```

Algorithm 6.9: Alternative implementation of Step 3 in TPLZC Main Encoding Procedure

6.6 Wavelet Transform for the LZC Algorithm

As discussed in Section 3.3.5, the filter set used for implementing the wavelet transform has a direct effect on the performance of image decomposition, which in turn affects the compression result. In addition, the number of transform levels used to decompose an image will affect the formation of the zerotree structure. Therefore, in this section we will discuss which wavelet filter set we should choose and how many decomposition levels we should use in the implementation of a wavelet transform for the LZC algorithm.

6.6.1 The Selection of Wavelet Filter

As mentioned previously in Section 3.3.5, for image compression the biorthogonal filter is more appropriate than the orthogonal filter. Therefore, the only question left is which biorthogonal filter set we should use for LZC.

There are two commonly used biorthogonal filter sets: the 9/7 filter set [89] and the 5/3 filter set [179, 180, 181, 182]. Because of their superior performance, these two filter set have also been chosen for the new JPEG 2000 standard [183, 184, 185]. Therefore, for comparison purposes, we will use the 9/7 and 5/3 filter sets for the LZC algorithm implementation.

The 9/7 filter set is a floating point filter set, so it is only suitable for lossy image compression. However, the 5/3 filter set is an integer filter set which is designed for reversible transformation, so it is suitable for both lossy and lossless image compression. The tap coefficients of the 9/7 and 5/3 filter sets are given in Table 6.1 and Table 6.2, respectively. The tap coefficients of the analysis filter pair H and G are denoted as $h(n)$ and $g(n)$, and the tab coefficients of the synthesis filter pair $\tilde{H}$ and $\tilde{G}$ are denoted as $\tilde{h}(n)$ and $\tilde{g}(n)$. The number n in the table is the tap index.

		Analysis Filter Coefficients		Synthesis Filter Coefficients	
n	factor	LPF H, $h(n)$	HPF G, $g(n)$	LPF $\tilde{H}$, $\tilde{h}(n)$	HPF $\tilde{G}$, $\tilde{g}(n)$
0	$\sqrt{2}$	0.602949	0.557543	0.557543	0.602949
± 1	$\sqrt{2}$	0.266864	-0.295636	0.295636	-0.266864
± 2	$\sqrt{2}$	-0.078223	-0.028772	-0.028772	-0.078223
± 3	$\sqrt{2}$	-0.016864	0.045636	-0.045636	0.016864
± 4	$\sqrt{2}$	0.026749			0.026749

Table 6.1: 9/7 biorthogonal filter coefficients

		Analysis Filter Coefficients		Synthesis Filter Coefficients	
n	factor	LPF H, $h(n)$	HPF G, $g(n)$	LPF $\tilde{H}$, $\tilde{h}(n)$	HPF $\tilde{G}$, $\tilde{g}(n)$
0	$\sqrt{2}$	0.75	0.5	0.5	0.75
±1	$\sqrt{2}$	0.25	-0.25	0.25	-0.25
±2	$\sqrt{2}$	-0.125			-0.125

Table 6.2: 5/3 biorthogonal filter coefficients

Although the analysis high-pass filter G is symmetric at index 0, according to Equation (3.30) in Section 3.3, the high-pass filter G should lag the low-pass filter H by one sample in the forward wavelet transform. Likewise, according to Equation (3.31) in Section 3.3, the synthesis high-pass filter $\tilde{G}$ should lead the synthesis low-pass filter $\tilde{H}$ by one sample in the inverse wavelet transform. Furthermore, for perfect reconstruction (PR), the analysis and synthesis filter pairs have to satisfy the following conditions [91](ch.4):

$$\text{No distortion} \qquad H(z)\tilde{H}(z) + G(z)\tilde{G}(z) = 2z^{-l}, \tag{6.8}$$

$$\text{Alias cancellation} \qquad H(-z)\tilde{H}(z) + G(-z)\tilde{G}(z) = 0, \tag{6.9}$$

$$\tilde{H}(z) = G(-z), \tag{6.10}$$

$$\tilde{G}(z) = -H(-z). \tag{6.11}$$

Therefore, based on perfect reconstruction conditions, filter tap coefficients should be multiplied by a normalization factor, $\sqrt{2}$. For the 9/7 filter set, multiplying by a factor of $\sqrt{2}$ does not have any impact on the system's computational complexity and memory requirement, as the 9/7 filter set originally requires floating point computation. However, multiplying the tab coefficients of 5/3 filter set by $\sqrt{2}$ will adversely increase the system computational complexity, by shifting from integer to floating point operation. The solution for this problem is to multiply either H and $\tilde{G}$, or G and $\tilde{H}$ by 2 instead of multiplying H, G, $\tilde{H}$, and $\tilde{G}$ by $\sqrt{2}$. However, multiplying filter coefficients by $\sqrt{2}$ or 2 will affect the magnitude dynamic range of the wavelet subbands and in turn may affect the performance of the LZC algorithm.

Figure 6.18 illustrates three variations of the magnitude dynamic range corresponding to the multiplication by factor of $\sqrt{2}$ or 2. In Figure 6.18(a), filter coefficients are multiplied by $\sqrt{2}$. The number marked on each subband is the *multiple* of the magnitude dynamic range. For example, if the pixel magnitude of an image pixel ranges from 0 to 255, then after three level wavelet transforms the magnitude of the *LL* subband coefficients will range from 0 to 2047, as the multiple is '8' (left-shift by 3 digits). We can see that in this case the increase of the dynamic range matches the importance of the wavelet subbands well and satisfies the working criterion for progressive bit-layer coding performed by LZC. Also, this is the most widely used wavelet transform case for zerotree coding algorithms.

In Figure 6.18(b), H and $\tilde{G}$ are multiplied by 2. Although the dynamic range increases dramatically, it still reflects the importance of the subbands and forms the working criterion for progressive bit-layer coding. Therefore, the wavelet matrix can still be coded by the LZC algorithm.

In Figure 6.18(c), G and $\tilde{H}$ are multiplied by 2. In this case, the increase of subband dynamic range no longer reflects the importance of the wavelet subbands. That is, the most important *LL* subband has the smallest dynamic range, but the least important *HH* subbands have the largest dynamic ranges. For bit-layer coding, coefficients with larger magnitudes always get more bit-budget allocation. Consequently, if this wavelet matrix is coded by LZC, the LZC codec will fail to correctly allocate bit-budget to encode important wavelet coefficients.

In short, if a wavelet transform is performed using floating point filters then applying the normalization factor, $\sqrt{2}$, to all analysis and synthesis filters will be a better choice, since it gives a more graceful increase of wavelet subband dynamic range. However, if the wavelet transform is performed using integer operations as when using 5/3 filters, then in order to ensure the validity of the LZC algorithm the normalization factor 2 should only be applied to H and $\tilde{G}$, not to $\tilde{H}$ and G.

6.6.2 The Selection of Wavelet Decomposition Level

The wavelet transform is used to reduce image spatial redundancy for a more compact image data representation, so intuitively the higher the number of the wavelet decomposition levels, the better the decorrelation and the more compact the image data.

Moreover, having more wavelet decomposition levels can enable more coefficients to be grouped into a coefficient tree. This results in a more effective zerotree structure, because a single zerotree symbol will be able to encode more insignificant coefficients. For instance, Figure 6.19 shows that four separate wavelet coefficient trees on a 3-level decomposed wavelet matrix will need to be coded by four zerotree roots. However, if we decompose the wavelet matrix by one more level, those four trees can then be grouped into a single tree and can be coded by one zerotree root. Consequently, the coding bit-budget can be saved to code other more important information.

However, the number of wavelet decomposition levels is limited by three factors: the image size, the filter size, and the system bus width. Firstly, a larger image generally can be decomposed into more wavelet transform levels than a smaller image, but depending on the image size the number of decompositions available may vary. This is because an image dimension must be dyadic in order to apply a wavelet transform to it. Therefore, the number of decom-

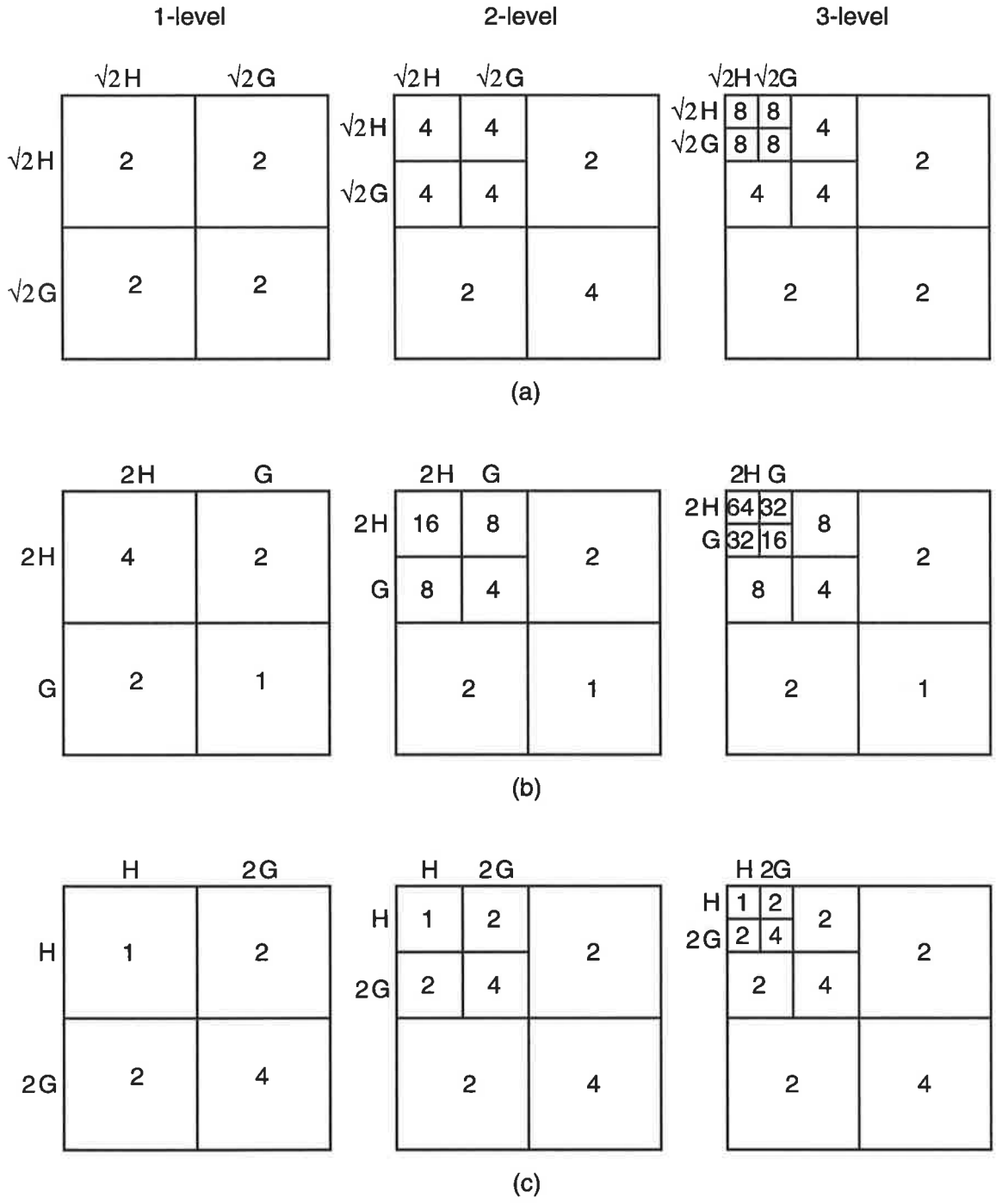


Figure 6.18: The variations of wavelet subband dynamic range (a) multiplying G and H by $\sqrt{2}$ (b) multiplying H by 2 (c) multiplying G by 2

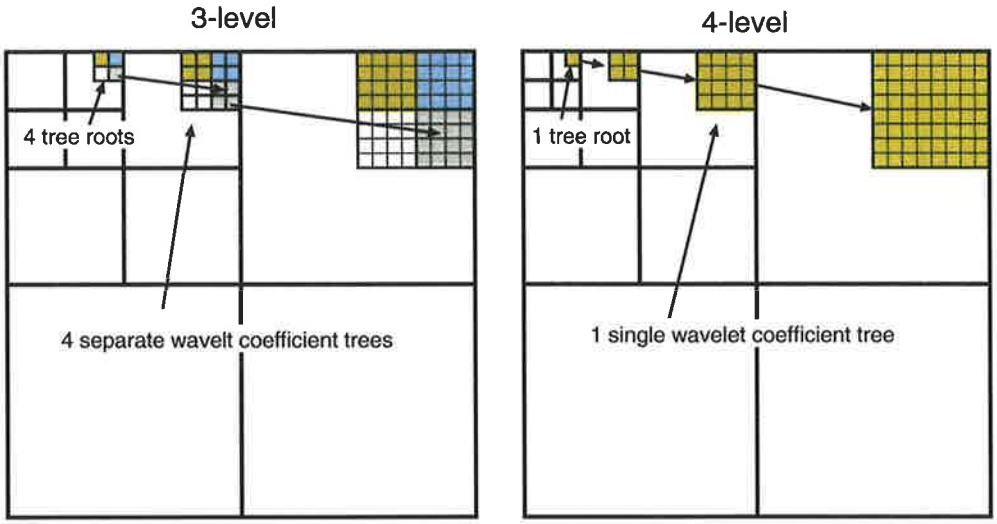


Figure 6.19: The coefficient tree formation on the wavelet matrix with different number of transform levels

position levels available will depend on how many times 2 can be factored out from the image dimension.

Secondly, every wavelet filter set theoretically can be used to decompose an image to the coarsest level, ie. only one coefficient in the *LL* subband, provided the image dimension is dyadic. However, in practice, as the dimension of the *LL* subband gets smaller, coefficient padding will become a dominant issue, especially for the symmetric extension method. An improper coefficient padding on the *LL* subband may cause a catastrophic reconstructed result.

Thirdly, as mentioned earlier, the magnitude dynamic range of wavelet coefficients will increase with the number of transform levels. As a result, the magnitude dynamic range may exceed the system bus width, and the wavelet coefficients may not be processed accurately.

In short, the number of wavelet decomposition levels can be increased by adjusting the image dimension, such as by padding extra coefficients on the edge, by using a shorter filter for higher level decomposition, or by increasing bus width. Nevertheless, those adjustments can also increase the coding complexity, as well as the implementation cost. There is no fixed rule to choose the best number of wavelet decomposition levels. But for the best reconstructed image quality, we may presume the number of decomposition levels should be five or six to match the number of frequency channels in the human visual system [58] as discussed in Section 2.4. Also, to ensure a proper formation of the LZC tree structure, the minimum number of decomposition levels has to be at least three, regardless of the image size.

6.7 The Test Codec for the LZC algorithm

This section briefly discusses the test codec used to verify the performance of the LZC algorithm for color image compression. The LZC test codec was implemented in the ANSI C Language, although one of the motivations for deriving the LZC algorithm is for hardware implementation.

The test codec has four modules for both the encoding and the decoding process. Those four modules are used for color space transform, wavelet transform, LZC or TPLZC, and arithmetic coding. The block diagram of the LZC test codec is shown in Figure 6.20

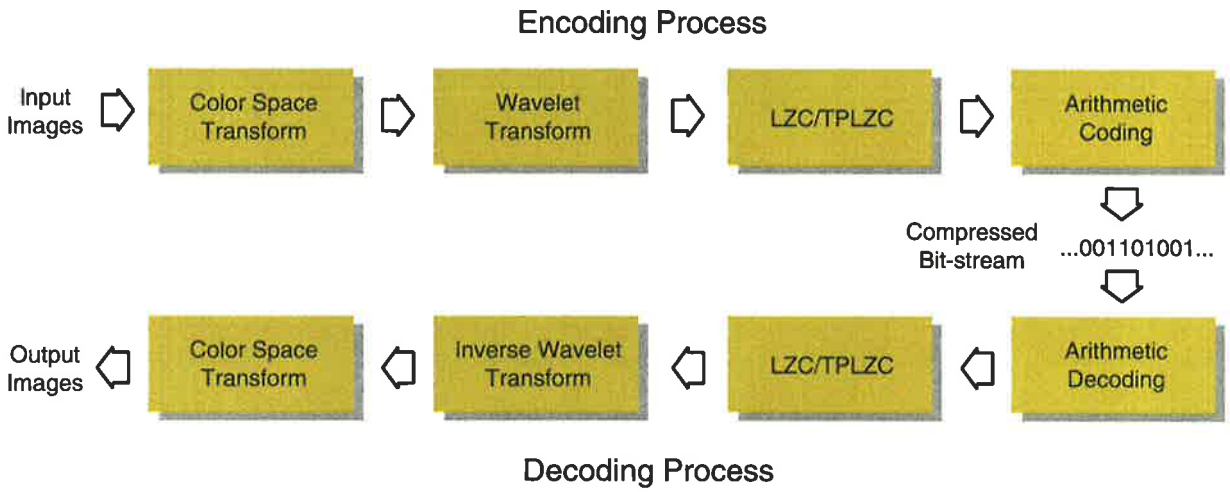


Figure 6.20: Different Modules of the LZC test codec

The performance of the test codec mainly depends on the LZC/TPLZC module, which implements the coding procedures of the LZC and TPLZC algorithms discussed in Sections 6.4 and 6.5. The color space transform module and wavelet transform module provide some freedom to select different coding options, including different color spaces, different wavelet filter sets, different signal extension methods, and different wavelet decomposition levels. Therefore, the performance of the test codec may fluctuate depending on the options chosen. The bit-stream generated by the LZC/TPLZC module is further arithmetic coded to reduce the entropy by using Witten et al.'s method [107]. That is, the output bits from the LZC/TPLZC module are grouped into bytes to form the coding symbols which are then fed into the arithmetic coding module for entropy coding. The probability model of the arithmetic coding is adaptively updated according to the coding symbols received and therefore is more appropriate for coding the visual data than a fixed model, such as discussed in Section 3.4.2.

6.8 Preliminary Experiments

As discussed in the previous section, we can choose different options for the LZC codec, including different color spaces, different wavelet filter sets, different signal extension methods, and different wavelet decomposition levels. Therefore, in this section, we will examine those coding options by performing some experiments to see how those options affect the performance of the LZC algorithm. Those coding options will have the same effect on both the LZC and TPLZC algorithm, so for brevity, the results shown in this section are those obtained from the LZC test codec only.

6.8.1 Color Space Experiment

Although the color coding sequence of the LZC algorithm is optimized for the YUV or YCbCr color spaces, there is no restriction that the LZC codec cannot perform coding in the RGB color space. However, as we discussed in Section 3.2, RGB components are highly correlated, so the LZC codec may not be able to effectively allocate bit-budget to some important information.

Therefore, theoretically the LZC codec will have a better performance in the YUV or YCbCr color spaces than in the RGB color space. This is because transforming images from RGB to YUV or YCbCr color space not only reduces the color information correlation, but also results in a more compact color data representation that matches the characteristics of the HVS. To verify this presumption, the LZC codec was tested in RGB, YUV and YCbCr color spaces. The other coding options used were fixed to: using 9/7 wavelet filter set; symmetric image extension; and five or six wavelet decomposition levels depending on the image dimensions.

The test LZC codec was used to compress some test images in RGB, YUV and YCbCr color spaces at a variety of coding bit-rates. The PSNR results from those images are presented in Figure 6.21. The results show that the coding performance of the LZC codec is substantially improved by transforming images from RGB to either YUV or YCbCr color space, which confirms our presumption. However, compressing the *Flowerfield1* image in YUV and YCbCr spaces only gives slightly better PSNRs than in RGB space, and compressing the *Mt Fuji3* image in RGB space even appears to have better results than in YUV and YCbCr spaces at some bit-rates. This may due to the RGB color coding order performed by the codec coincidentally matching the order of information importance for that particular image.

Nevertheless, the chance that compressing images in the RGB space gives better compression results is rare. Generally, performing the LZC algorithm in both YUV and YCbCr color spaces will give a better compression result than in RGB space. Hence, it is worthwhile to spend some extra complexity to transform images from RGB to either the YUV or the YCbCr color space.

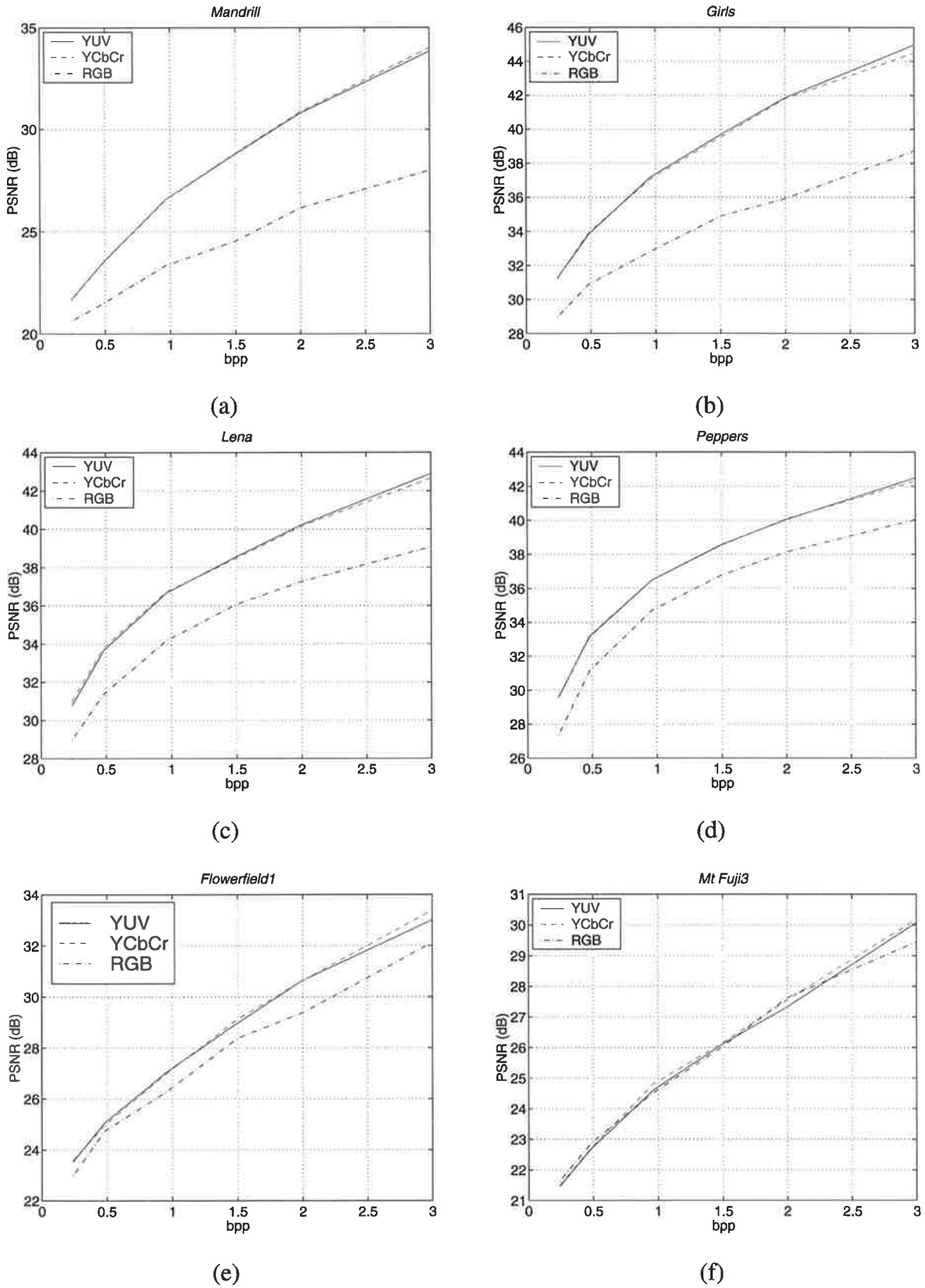


Figure 6.21: Performance comparison by using RGB, YUV, or YCbCr color space
(a) *Mandrill* (b) *Girls* (c) *Lena* (d) *Peppers* (e) *Flowerfield1* (f) *Mt Fuji3*

6.8.2 Wavelet Filter Experiment

The wavelet filter set used to decompose images has a direct effect on the coding performance of the LZC algorithm, because different filter sets have different performance in decorrelating images. The LZC codec will mainly use the 9/7 filter set for the wavelet transform. However, as the tap coefficients of the 5/3 filter set are integers, it is suitable for lossless compression or hardware implementation.

In order to find out how much the performance difference exists between using 9/7 and 5/3 filter sets for wavelet decomposition, these two filter sets were used in the LZC test codec to compress several images at a variety of bit-rates. The other coding options were fixed to YUV color space, symmetric extension, and five or six wavelet transform level depending on the image size.

The experimental results in Figure 6.22 show that in terms of PSNR values the LZC codec generally performs better when using the 9/7 filter set. However, the PSNR difference is marginal, about 0.5 dB in average.

Therefore, the choice of using 9/7 or 5/3 filters depends on the area of application. For example, if the LZC algorithm is used for software implementation or for lossy compression, then the 9/7 filter is a better choice. On the other hand, if LZC algorithm is applied to hardware implementation or for lossless compression, like for medical images, then the 5/3 filter will be a more appropriate choice.

6.8.3 Signal Extension Experiment

As discussed in Section 3.3.5, we can use either periodic or symmetric extension to pad coefficients at the image boundary for the wavelet transform. However, under a severe quantization, the reconstructed image padded by the periodic method will have some unpleasant artifacts at the boundary due to discontinuity. On the other hand, the image padded by the symmetric method will give a smooth boundary, even when severely quantized. Therefore, theoretically, the LZC test codec should use the symmetric method in the wavelet transform for better compression performance.

The performance difference between using symmetric and periodic extensions for the LZC algorithm is examined here to verify this presumption. The coding option is varied between using symmetric and periodic extension, but fixed for using YUV color space, 9/7 filter set, and five or six wavelet transform levels.

Figure 6.23 shows the PSNR values from coding some test images at a variety of bit-rates. The results show that using symmetric extension gives about 0.25 dB improvement in the recon-

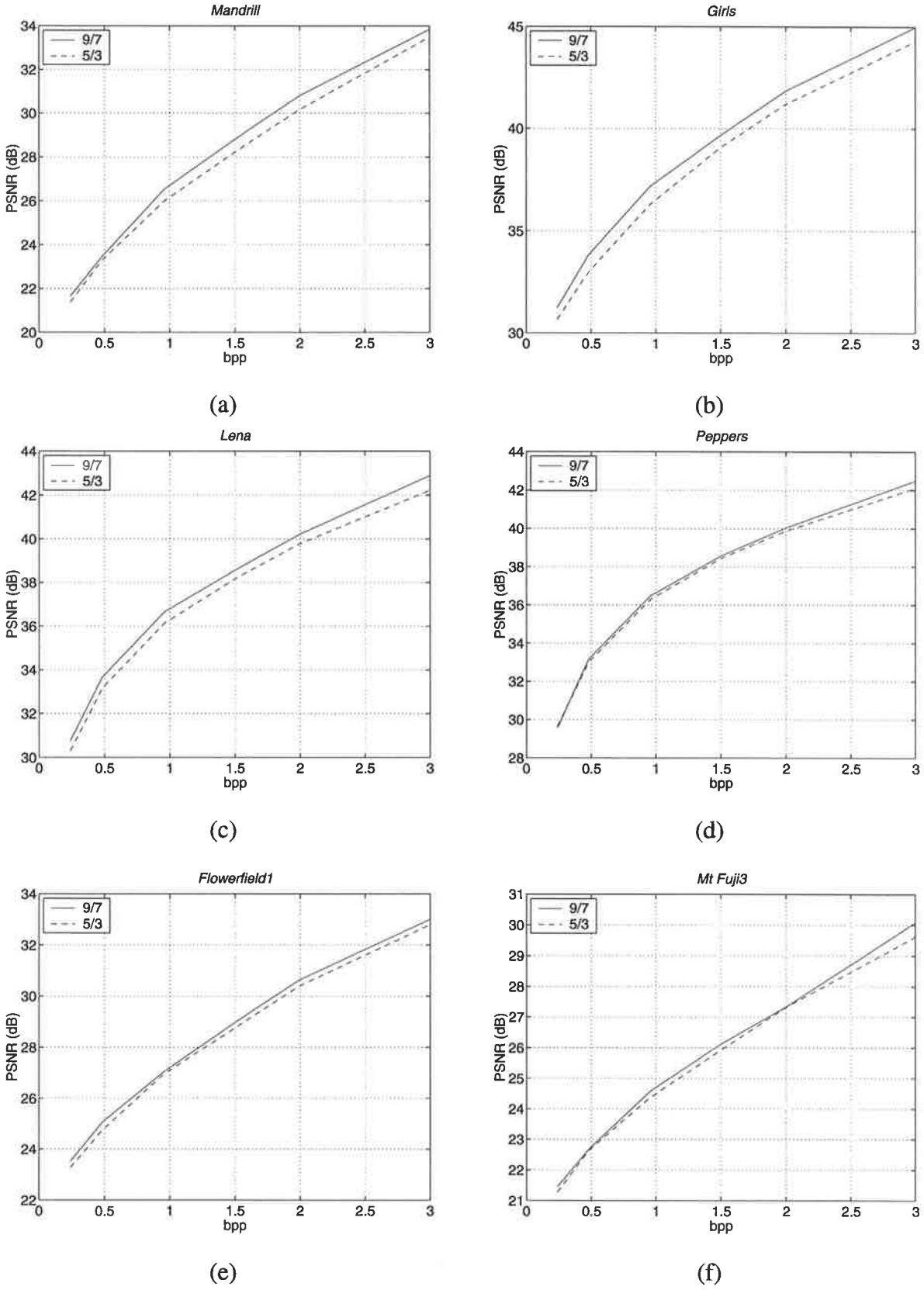


Figure 6.22: Performance comparison by using 9/7 or 5/3 wavelet filter set (a) *Mandrill* (b) *Girls* (c) *Lena* (d) *Peppers* (e) *Flowerfield1* (f) *Mt Fuji3*

struction image quality. For some images, symmetric extension only gives a marginal improvement in the PSNR values, in particular for those having a lot of high frequency features, such as *Mandrill* and *Pot*. However, despite the improvement of only about 0.25 dB, the subjective comparison shows that under severe quantization, the image coded with symmetric extension will have a better visual quality. For instance, when the *Flowerfield1* image is coded at 0.24 bpp, the symmetric extended version in Figure 6.24(a) shows a very *clean* sky in the background. While the periodic extended version in Figure 6.24(b) shows some unpleasant artifacts in the sky, due to the discontinuity from wrapping the red flower field to the blue sky.

In fact, the quality difference between using symmetric and periodic extensions is due to quantization errors [91](p.340). If we increase the coding bit-rate and let the codec have more bit-budget to code more bit-layers, the visual quality difference between using these two extension methods will decrease. For example, when *Flowerfield1* image is coded at 0.96 bpp, the visual quality difference between using symmetric and periodic extension methods has reduced dramatically, as shown in Figure 6.24(c) and (d). As we keep increasing the coding bit-rate, the perceived quantization errors will eventually drop below the *JND* level and there will be no visual quality difference between using these two extension methods.

Nevertheless, for all bit-rates, the symmetric extension method generally results in better objective and subjective reconstructed quality, so it is used for the LZC test codec.

6.8.4 Wavelet Decomposition Level Experiment

The number of wavelet decomposition levels directly affects the formation of LZC coefficient trees and, therefore, affects the reconstructed image quality. In Section 6.6.2, we presumed that the number of wavelet decomposition levels should be five or six following the number of frequency channels in the human visual system. In addition, we also presumed that the minimum number of wavelet decomposition levels should be three for an effective formation of LZC coefficient trees.

In this experiment, we verify our presumptions by varying the decomposition level coding option in the LZC test codec to compress several test images. The number of decomposition levels ranges from two to seven for the *Mandrill*, *Girls*, and *Peppers* images, two to six for the *Cacti* images, and two to five for the *Pot* and *Mt Fuji1* image. The other coding options were fixed to YUV color space, 9/7 filter set, and symmetric extension.

The results shown in Figure 6.25 are for the coding bit-rates of 0.48 bpp and 0.96 bpp. The results show that, for both bit-rates, increasing the number of decomposition levels from two to three improves the reconstructed quality dramatically. This is because more coefficients are grouped in a LZC tree in a 3-level decomposed wavelet matrix than in a 2-level decomposed

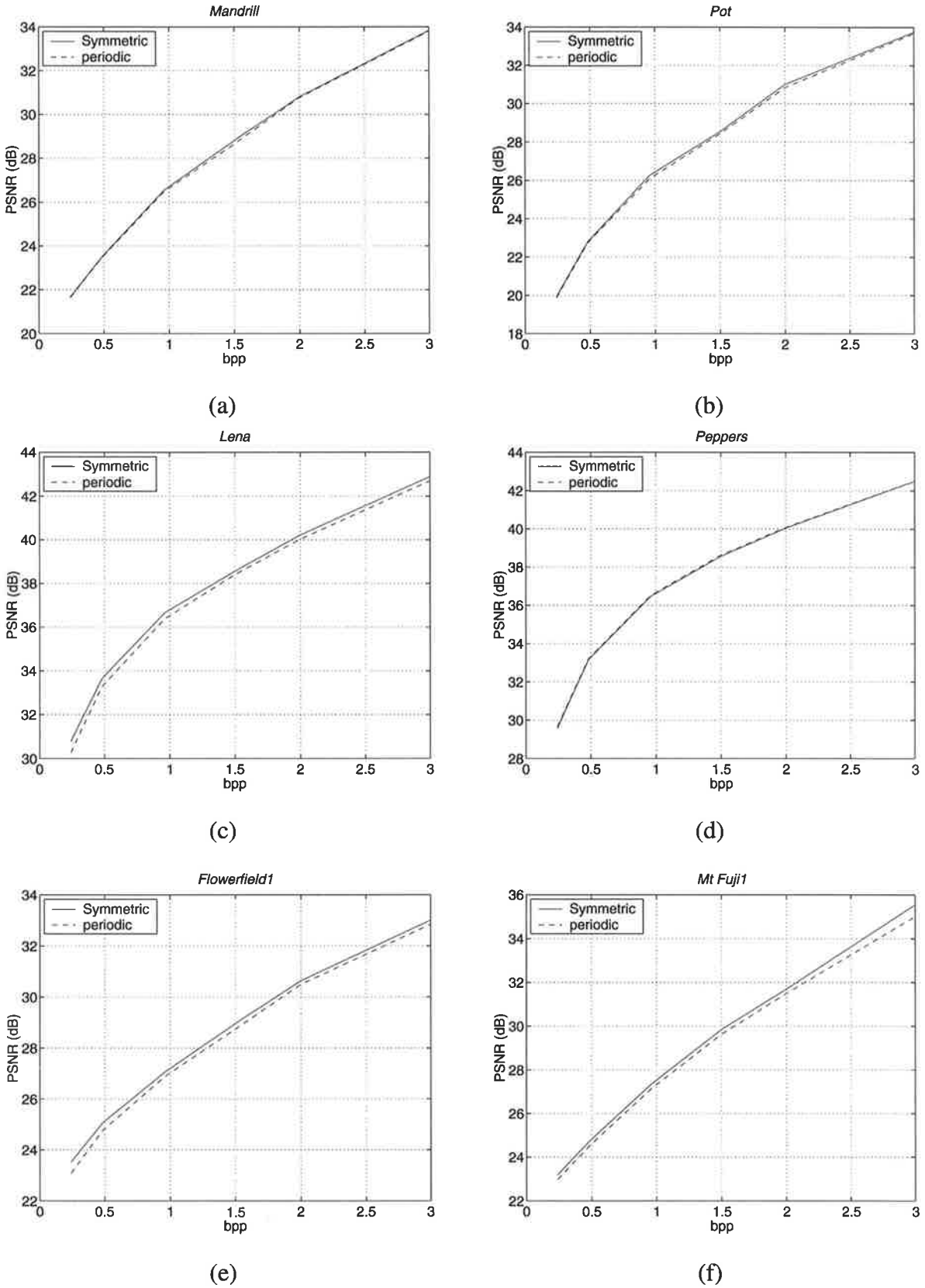


Figure 6.23: Performance comparison by using symmetric or periodic extension (a) *Mandrill* (b) *Pot* (c) *Lena* (d) *Peppers* (e) *Flowerfield1* (f) *Mt Fuji1*

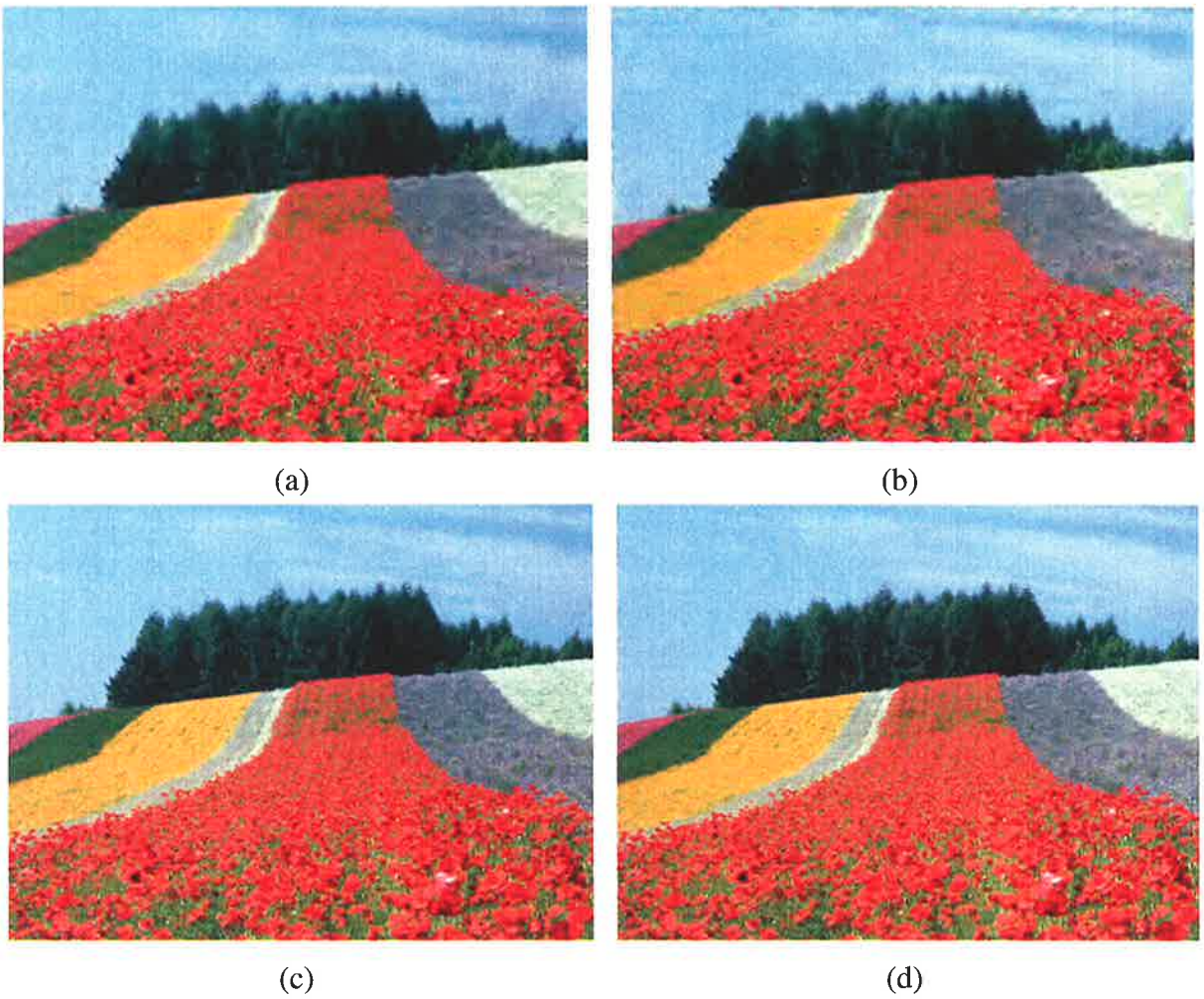


Figure 6.24: The *Flowerfield1* image (a) at 0.24 bpp by symmetric extension (b) at 0.24 bpp by periodic extension (c) at 0.96 bpp by symmetric extension (d) at 0.96 bpp by periodic extension

matrix. Therefore, more insignificant coefficients can be coded by a single zerotree root. This verifies the presumption for the minimum number of decomposition levels required for good formation of LZC coefficient trees.

The quality continues to improve as we increase the number of decomposition levels from three to four, but the improvement becomes not so significant. The improvement on the reconstructed quality is stabilized at around five or six decomposition levels, resulting in the best performance for the test codec. Above six levels, the quality improvement is only marginal. This is because each tree branch contains too many coefficients and covers a large spatial area of the image. Therefore, in order to form zerotrees, each tree branch will still need to be further partitioned. This verifies our presumption to use the number of frequency channels in the human visual system as the preferred decomposition level for the LZC test codec.

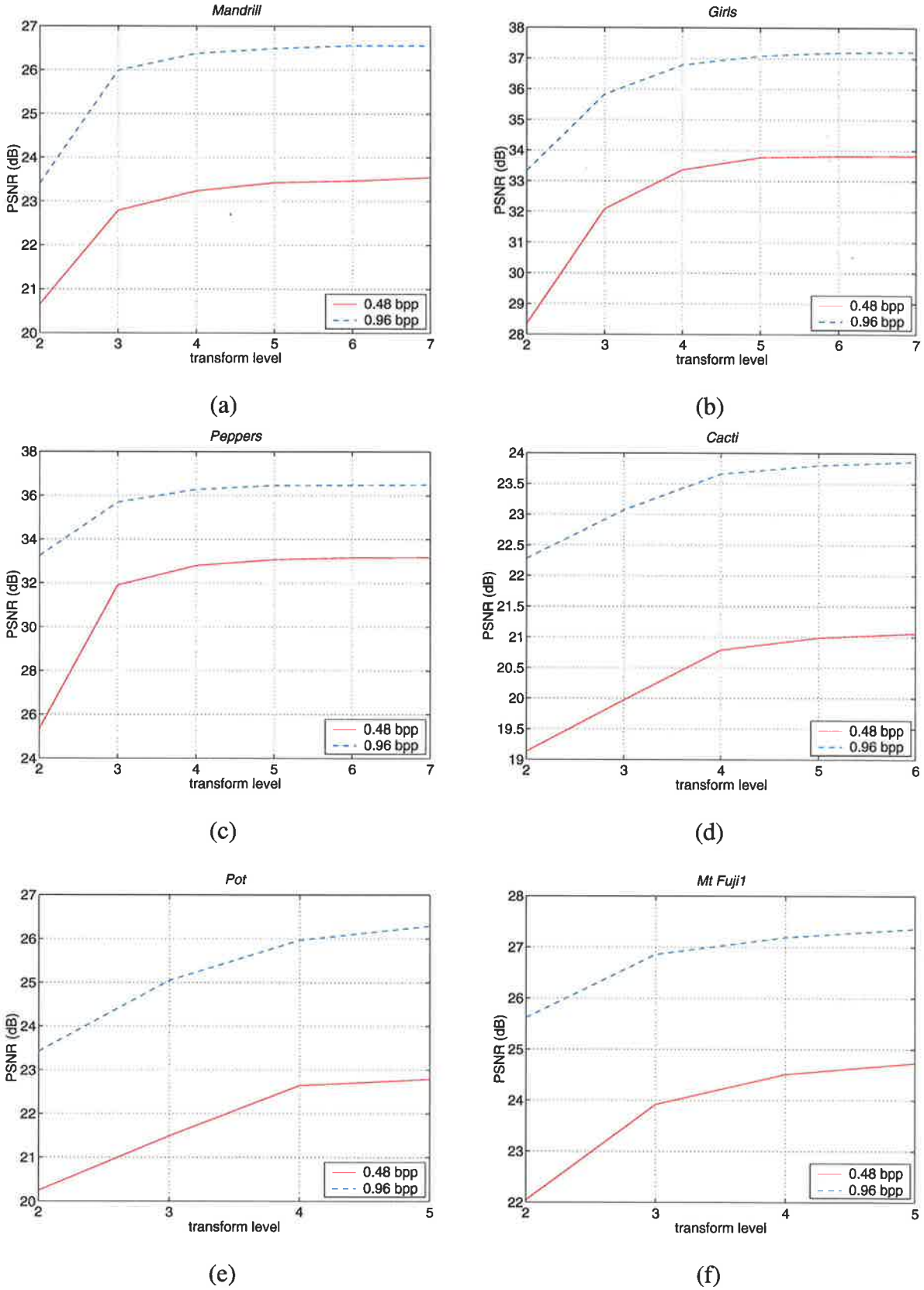


Figure 6.25: Performance comparison by using different wavelet transform levels (a) *Mandrill* (b) *Girls* (c) *Peppers* (d) *Cacti* (e) *Pot* (f) *Mt Fuji1*

6.9 Results

This section firstly overviews how the tree search methods affect the performance of the LZC algorithm by comparing some experimental results from the LZC and TPLZC test codecs. Secondly, results from the LZC test codec are compared with those from the SPIHT test codec for determining the performance of LZC algorithm. Finally, some results from the LZC test codec are compared with JPEG2000 results.

6.9.1 LZC and TPLZC

We have presumed that at low bit-rate, an image compressed by TPLZC will have a more uniform visual quality than compressed by LZC. To verify this presumption this section presents some results obtained from the LZC and TPLZC test codecs. The coding options used for both codecs were YUV color space, 9/7 wavelet filter set, symmetric extension, and five or six decomposition levels.

The images shown in Appendix A were used in the test, but only those showing noticeable visual differences are presented here. Figure 6.26 to 6.30 show the results in compressing *Mandrill*, *Matsuri*, *Mt Fuji2*, *Railway*, and *Flowers*. From those results we can see that images compressed by the TPLZC codec have more uniform visual quality, while those compressed by the LZC codec appear to have uneven visual quality, as the lower half looks blurry. The visual comparison of those results are summarized as follows.

- The compressed *Mandrill* images in Figure 6.26 show that at 0.24 bpp the TPLZC coded version has more details in hairs and whiskers; while those details in the lower half of the LZC coded version are very blurry. At 0.48 bpp, the TPLZC coded version has more details in face wrinkles and nose texture than the LZC coded version.

- The compressed *Matsuri* images in Figure 6.27 show that the lower half of the TPLZC coded version has more details of clothing texture and wrinkles, leg features, and the white traffic line on the ground; while those details in the LZC coded version are either missing or very blurry.

- The compressed *Mt Fuji2* images in Figure 6.28 show that the TPLZC coded version already has recognizable paddy field features, but those features in the LZC coded version is still not quite recognizable.

- The compressed *Railway* images in Figure 6.29 show that the TPLZC coded version has more details of the road texture, while those features in the LZC coded version still look

blurry.

- The compressed *Flowers* images in Figure 6.30 show that the TPLZC coded version has more leaf details, while leaf details in the LZC coded version are blurry.

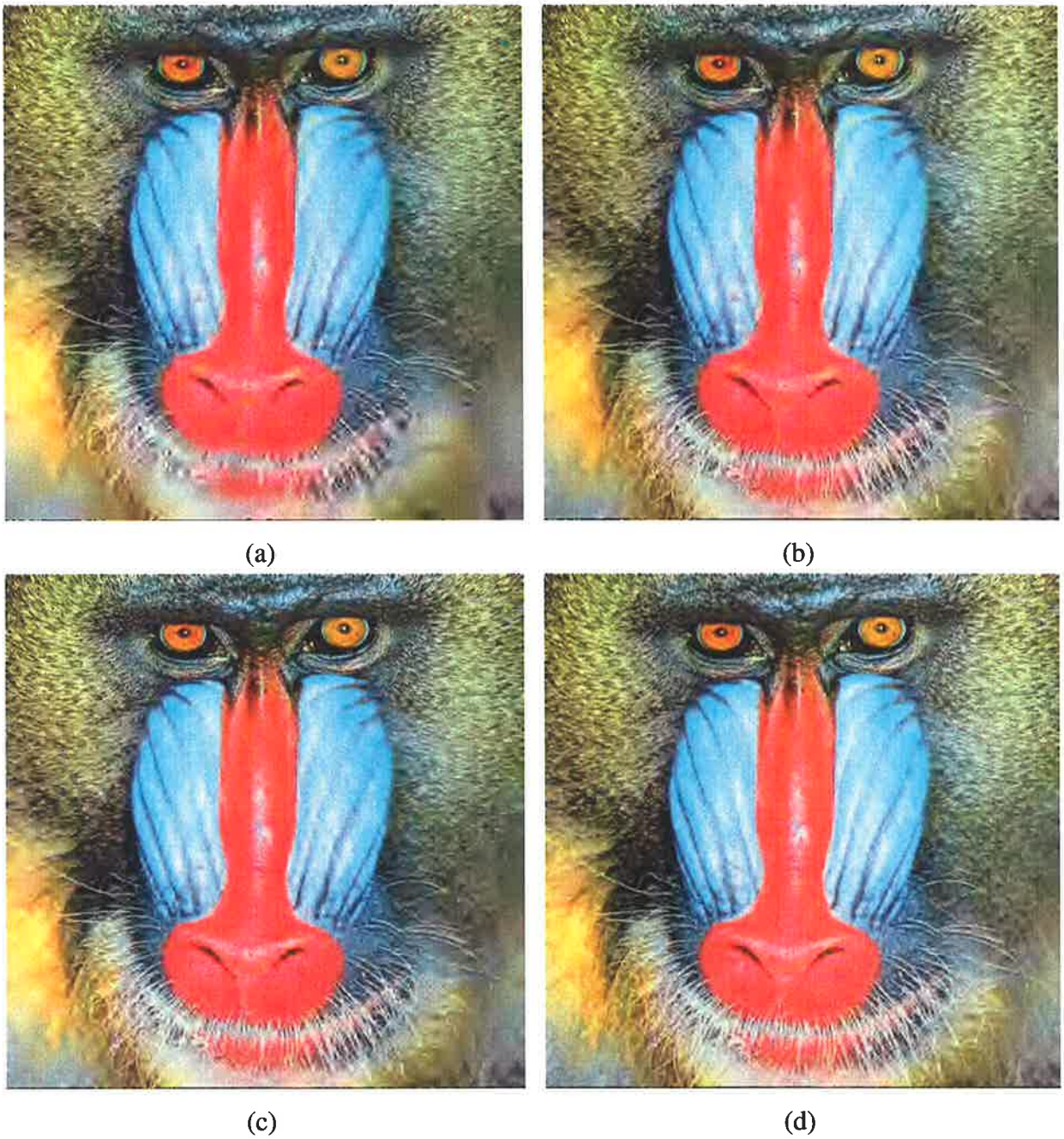
From the visual comparison discussed above, we can see how the recursive and raster tree search affect the reconstructed image quality. Images obtained from the LZC codec have better visual quality in the upper half, but worse visual quality in the lower half. This uneven visual quality on the reconstructed images is caused by the use of recursive tree search in the bit-allocation scheme, as we have pointed out in Section 6.5.1. This uneven visual quality is not visually pleasant if it is too noticeable, like that in the *Mandrill* image in Figure 6.26(a) and the *Matsuri* image in Figure 6.27(a).

In contrast, images obtained from the TPLZC codec have a more uniform visual quality and look more visually pleasant than the LZC coded versions. The uniform visual quality is the result of using the raster tree search for bit-allocation. That is, the TPLZC codec can allocate the bit-budget uniformly to code wavelet coefficients with the same importance rather than those with the same spatial locations.

Nevertheless, the visual quality difference between using LZC and TPLZC codec will decrease when the coding bit-rate increases. Eventually there will be no visual quality difference after increasing to a certain bit-rate, which enables quantization errors to fall below the JND level.

Figure 6.31 shows the PSNR values obtained for the test images shown in this section. Despite the visual quality comparison showing that TPLZC coded images generally have better visual quality than LZC coded ones, the PSNR comparison does not well reflect the visual quality perceived by the viewer. Only the *Mt Fuji2* PSNR results, shown in Figure 6.31(c), are consistent with the subjective comparison. The other PSNR results suggest that the LZC codec gives better compression quality than the TPLZC codec. Nevertheless, as discussed previously in Section 4.3, the PSNR quality measurement only provides a reference scale and might not predict the actual visual quality perceived by a human.

In short, based on the subjective comparison, the raster tree search is better than the recursive tree search at low coding bit-rate. Therefore, for low bit-rate applications, we may want to spend extra coding complexity and memory for implementing TPLZC, in order to obtain better visual quality in the reconstructed images.



(a) (b)

(c) (d)

Figure 6.26: The *Mandrill* image compressed by (a) LZC at 0.24bpp (b) TPLZC at 0.24bpp (c) LZC at 0.48bpp (d) TPLZC at 0.48bpp

(a)

(b)

Figure 6.27: The *Matsuri* image compressed by (a) LZC at 0.24bpp (b) TPLZC at 0.24bpp

(a)

(b)

Figure 6.28: The *Mt Fuji2* image compressed by (a) LZC at 0.24bpp (b) TPLZC at 0.24bpp

(a)

(b)

Figure 6.29: The *Railway* image compressed by (a) LZC at 0.24bpp (b) TPLZC at 0.24bpp

(a)

(b)

Figure 6.30: The *Flowers* image compressed by (a) LZC at 0.24bpp (b) TPLZC at 0.24bpp

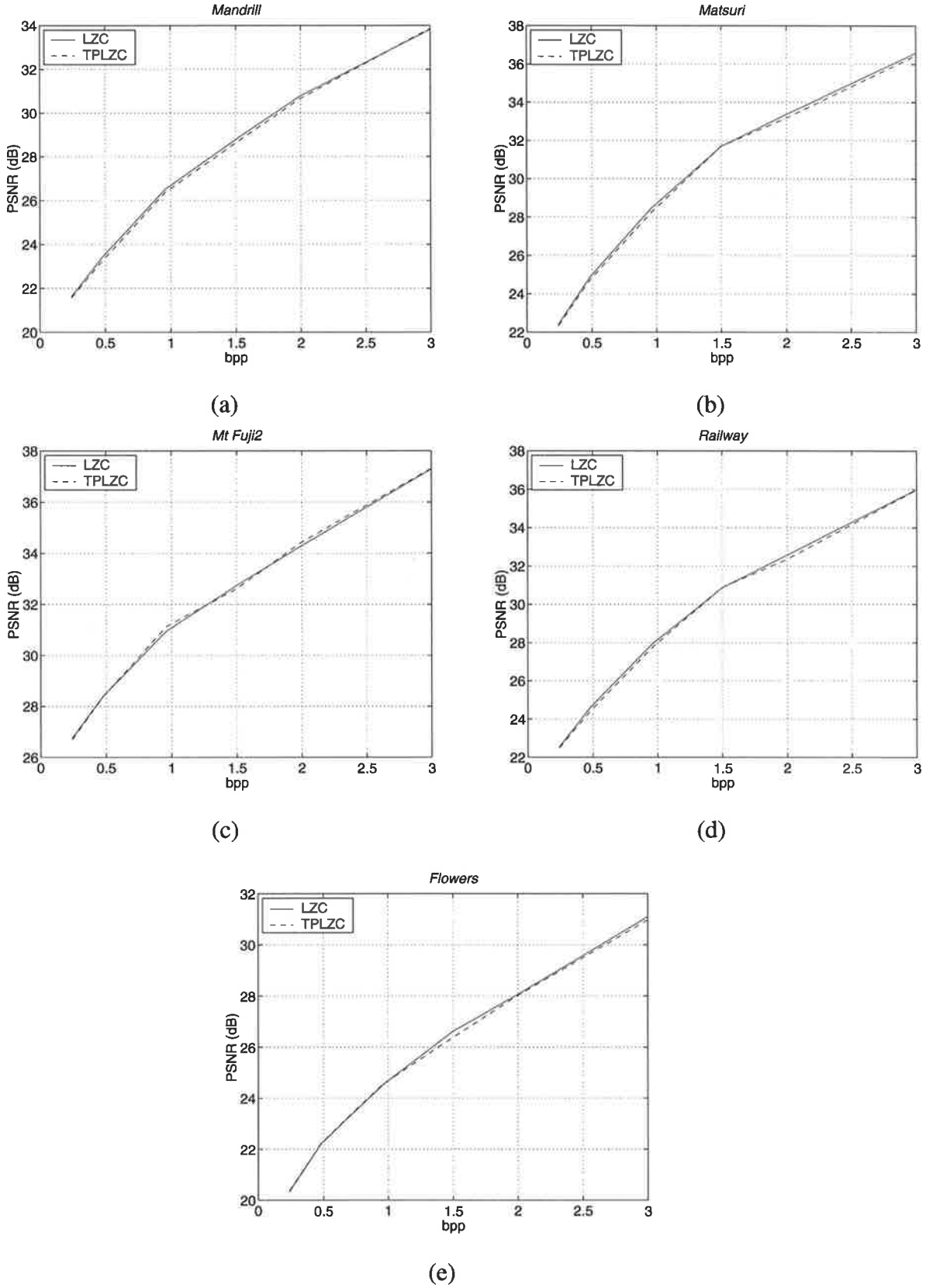


Figure 6.31: PSNR comparison - LZC and TPLZC (a) *Mandrill* (b) *Matsuri* (c) *Mt Fuji2* (d) *Railway* (e) *Flowers*

6.9.2 LZC and SPIHT

One of the motivations for deriving the LZC algorithm is to demonstrate that the SPIHT-like zerotree coding can be implemented at lower complexity and with a much lower coding memory requirement. Therefore, in this section, we will compare the results from LZC and TPLZC with those from SPIHT to see if the image quality is significantly affected by the removal of coding lists and reduction of coding complexity. The SPIHT test codec was downloaded from the SPIHT website - *http://www.cipr.rpi.edu/research/SPIHT/spiht3.html*. The coding options used for the LZC and TPLZC codecs were YUV color space, 9/7 wavelet filter set, symmetric extension, and five or six wavelet decomposition levels.

The compressed images shown in this section were those from LZC and SPIHT only. One exception is the *Mandrill* image, because the TPLZC compressed version has better subjective quality than the LZC compressed version.

Figures 6.32 to 6.35 present the visual quality comparison for the *Mandrill*, *Girls*, *Lena*, and *Peppers* images, which are the most commonly used test images. The results shown here are from those coded at 0.24 bpp and 0.48 bpp. From the visual comparison in those figures, we can hardly see any visual quality difference between using LZC and SPIHT to compress those images. For this reason the results at higher bit-rates are not shown here, because if we cannot see any visual difference at 0.24 bpp and 0.48 bpp, we will not be able to see any difference for images coded by LZC and SPIHT at higher bit-rates, at which the quantization errors will be close to or below the JND level of the HVS.

Figures 6.36 to 6.38 show some results from coding *Mt Fuji1*, *Flowerfield2*, and *Farm* images at 0.96 bpp. Those images are all natural scenery images with many high frequency features. The visual comparison for those images is summarized as follows.

- In Figure 6.36, the compressed *Mt Fuji1* images show that the LZC coded version has more details of Mt Fuji features, straw farm and tree features, and flower features. While those features on the SPIHT coded version are still very blurry, even at this bit-rate.
- In Figure 6.37, the compressed *Farm* images show that the LZC coded version has more details of the plowland features in the background than the SPIHT coded version.
- In Figure 6.38, the compressed *Flowerfield2* images show that the LZC coded version has more details of tree features in the background and the flower field feature in the foreground, while the same features on the SPIHT compressed version still look blurry.

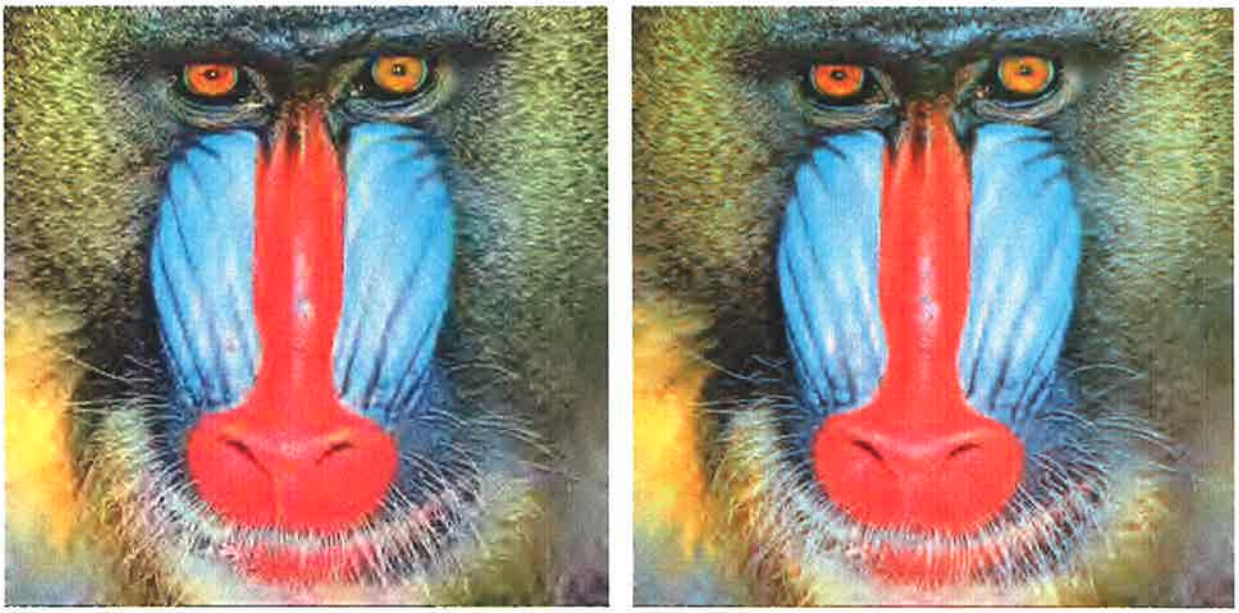
Interestingly, in spite of having lower coding memory and coding complexity, LZC can sometimes have better coding performance than SPIHT, especially for coding natural scenery

images with many high frequency features. This may be due to the fact that LZC is optimized to follow the characteristics of the HVS, while SPIHT is optimized to improve PSNR results [156].

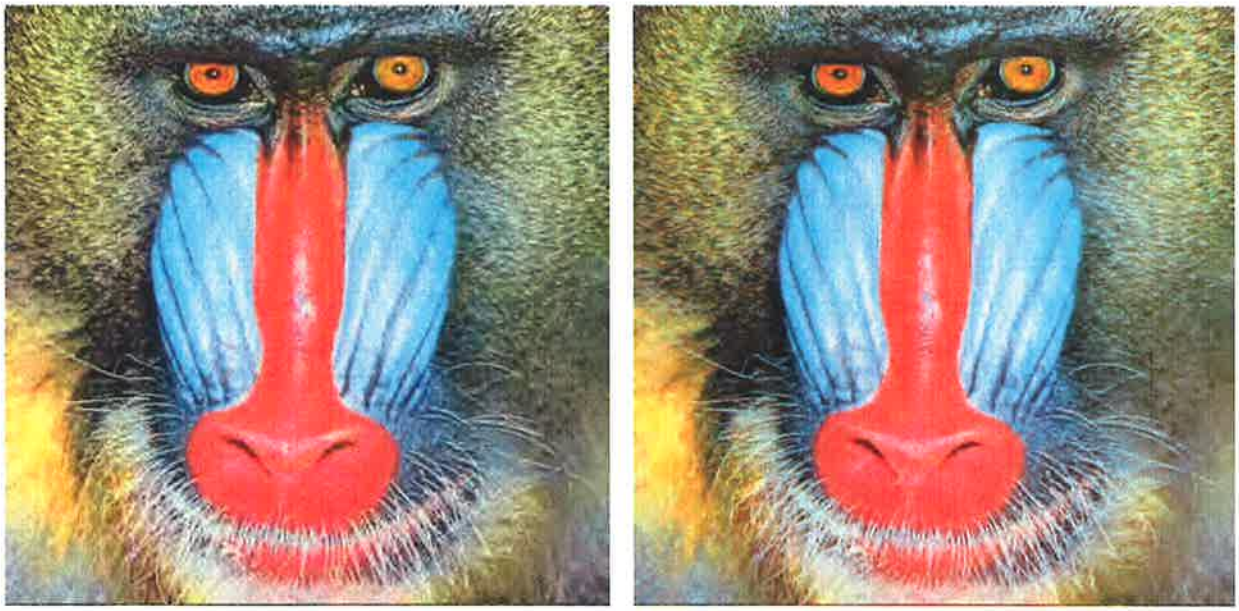
As a reference, Figure 6.39 and 6.40 give PSNR and RMSE comparisons for some test images compressed by LZC, TPLZC and SPIHT test codecs. Most of those test images have been shown for subjective comparison in this or the previous section, and some have not but can be found in Appendix A. From the PSNR and RMSE comparison, we can see that in most cases there is not much coding performance difference between using LZC, TPLZC and SPIHT in terms of objective quality. For some images, , such as *Farm* and *Flowerfield2*, SPIHT gives better objective quality performance than LZC. Nevertheless, despite showing better PSNRs, the SPIHT compressed *Farm* and *Flowerfield2* images actually show lower subjective quality than those compressed by LZC. This again verifies that PSNR and RMSE can only be used as a reference, but can not be used to predict relative visual quality.

The significance evaluation of the coefficient $C(i,j)$ and the descendant set $D(i,j)$ in LZC and TPLZC are exactly the same as in SPIHT. That is, the significance evaluation is done by recursively scanning down the tree branches. Although the significance evaluation may sound complicated and may require a lot of execution time, it actually benefits from the highly regular zerotree relations. Therefore, once the coordinates of the tree root are identified, the coordinates of the entire tree coefficients can then be derived by the zerotree relations discussed in Section 6.3. Furthermore, most of the significance evaluations of $C(i,j)$ and $D(i,j)$ are done in coding the first couple of bit-layers and the results are recorded on the F_C and F_D maps. Therefore, for coding the consecutive bit-layers the number of significance evaluations would be significantly reduced and would require only a fraction of the execution time of SPIHT. Moreover, the requirement of only one coding pass and only two bit-maps in LZC and TPLZC significantly lowers the number of memory accesses when compared to SPIHT. Hence, for hardware implementation, LZC and TPLZC would have a shorter execution time and a lower number of memory accesses than SPIHT.

In short, from the subjective comparison, the LZC algorithm demonstrates a similar coding performance to SPIHT for low bit-rate coding. For higher coding bit-rates, there is simply no performance difference between LZC and SPIHT in terms of visual quality. Therefore, the LZC algorithm can provide an alternative implementation of zerotree coding and achieves similar performance to SPIHT, but with lower coding memory requirement and lower coding complexity. Although some objective comparison shows SPIHT performs better than LZC, the subjective quality is really what concerns us.

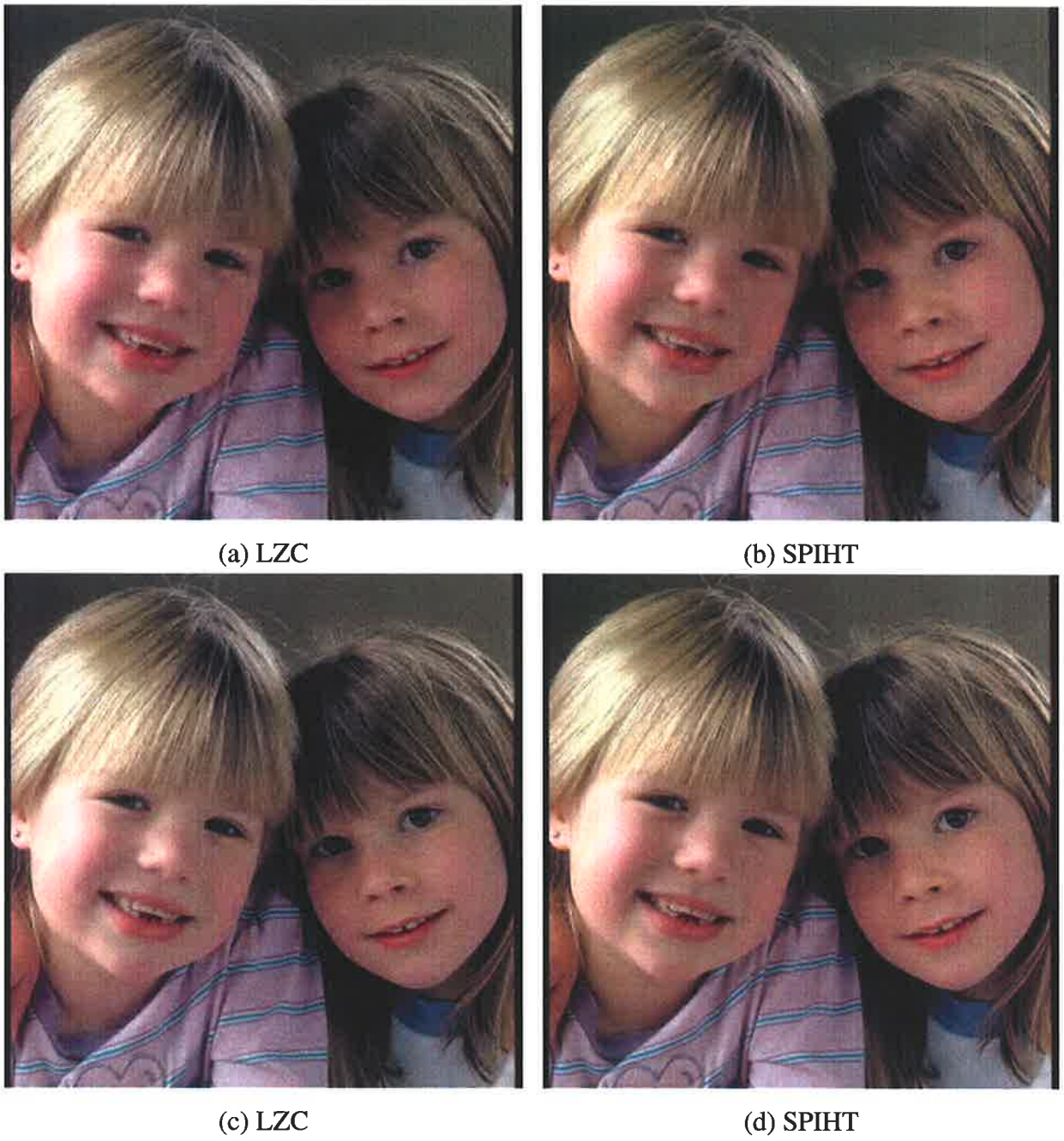


(a) TPLZC (b) SPIHT



(c) TPLZC (d) SPIHT

Figure 6.32: The *Mandrill* image compressed by (a) TPLZC at 0.24bpp (b) SPIHT at 0.24bpp (c) TPLZC at 0.48bpp (d) SPIHT at 0.48bpp



(a) LZC (b) SPIHT

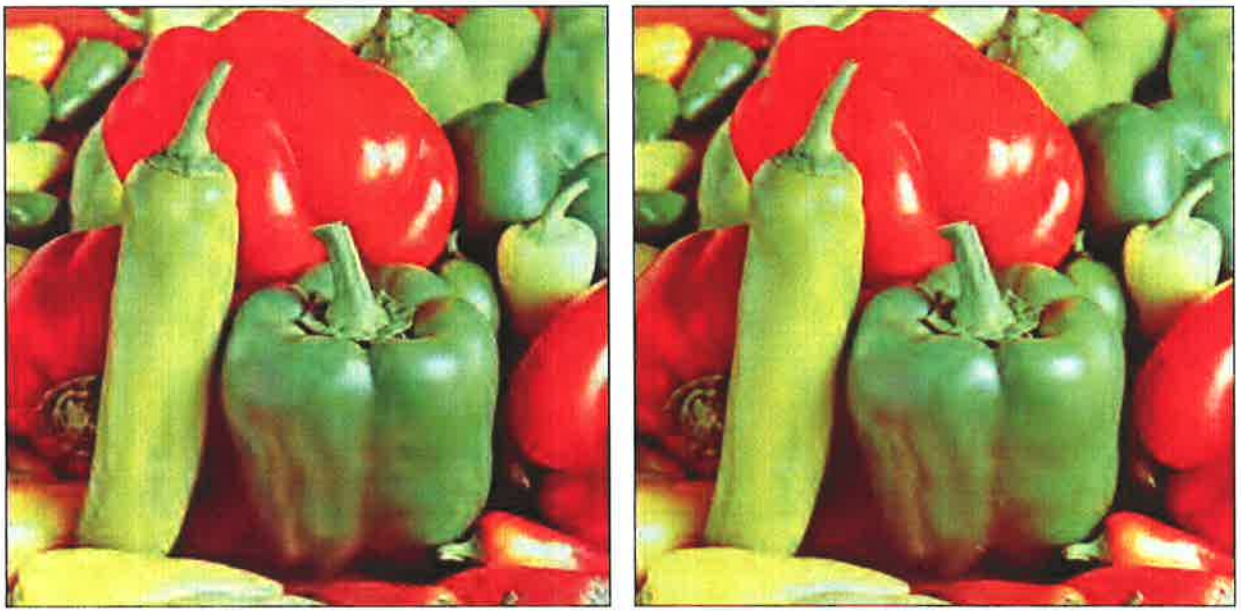
(c) LZC (d) SPIHT

Figure 6.33: The *Girls* image compressed by (a) LZC at 0.24bpp (b) SPIHT at 0.24bpp (c) LZC at 0.48bpp (d) SPIHT at 0.48bpp

(a) LZC (b) SPIHT

(c) LZC (d) SPIHT

Figure 6.34: The *Lena* image compressed by (a) LZC at 0.24bpp (b) SPIHT at 0.24bpp (c) LZC at 0.48bpp (d) SPIHT at 0.48bpp



(a) LZC (b) SPIHT

(c) LZC (d) SPIHT

Figure 6.35: The *Peppers* image compressed by (a) LZC at 0.24bpp (b) SPIHT at 0.24bpp (c) LZC at 0.48bpp (d) SPIHT at 0.48bpp

(a) LZC

(b) SPIHT

Figure 6.36: The *Mt Fuji1* image compressed at 0.96bpp by (a) LZC (b) SPIHT

(a) LZC

(b) SPIHT

Figure 6.37: The *Farm* image compressed at 0.96bpp by (a) LZC (b) SPIHT

(a) LZC

(b) SPIHT

Figure 6.38: The *Flowerfield2* image compressed at 0.96bpp by (a) LZC (b) SPIHT

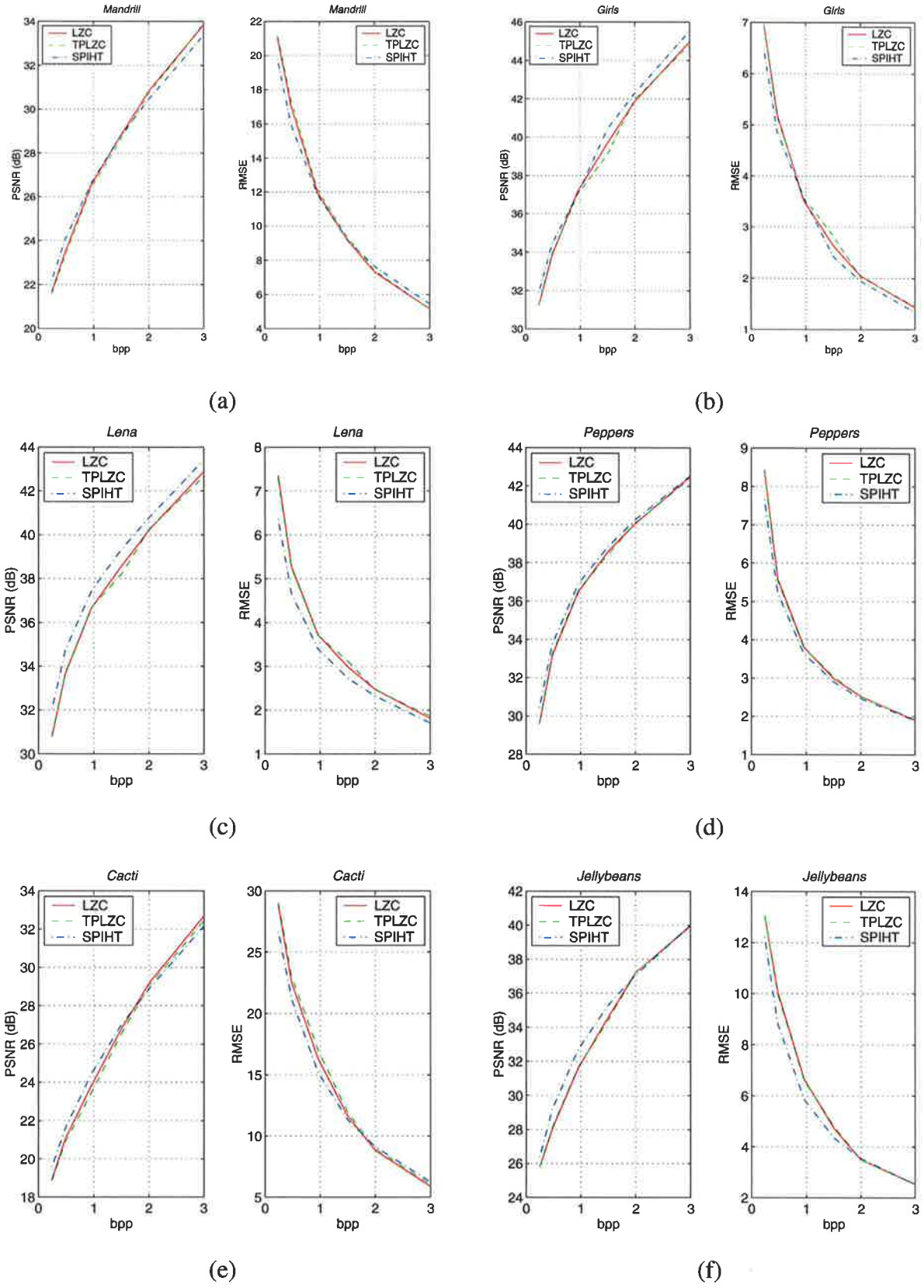


Figure 6.39: PSNR comparison - LZC, TPLZC, and SPIHT (a) *Mandrill* (b) *Girls* (c) *Lena* (d) *Peppers* (e) *Cacti* (f) *Jellybeans*

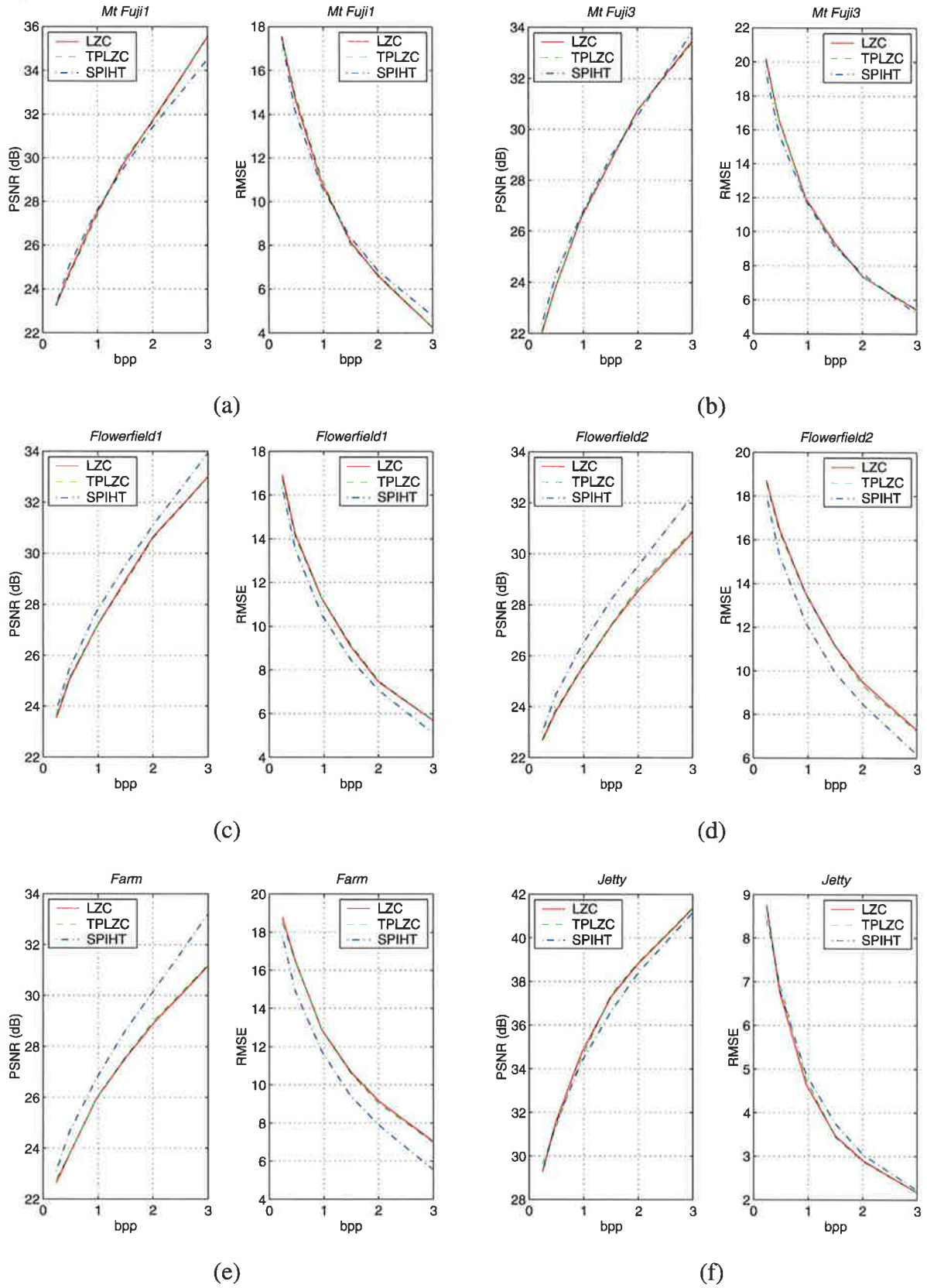
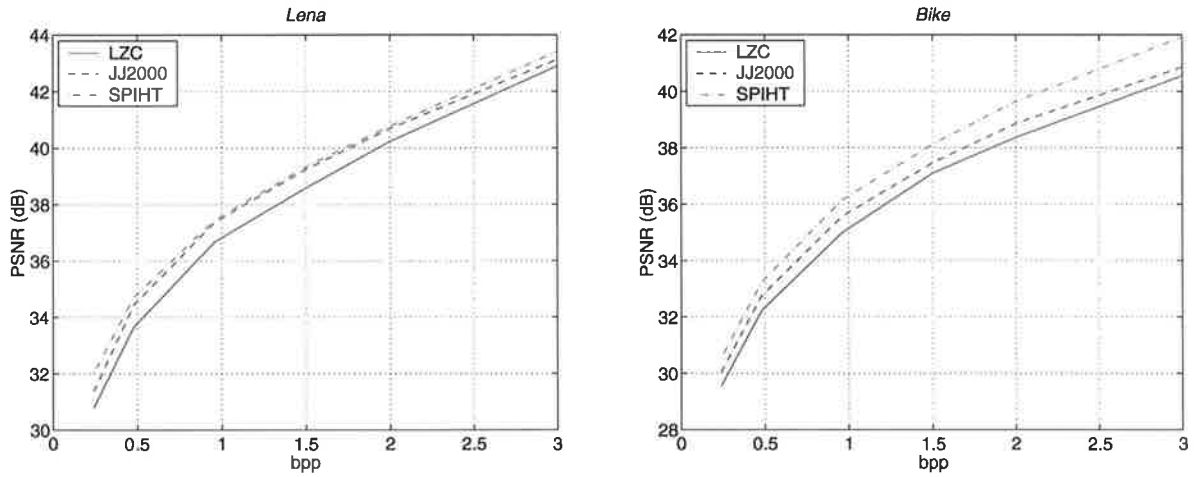


Figure 6.40: PSNR comparison - LZC, TPLZC, and SPIHT (a) *Mt Fuji1* (b) *Mt Fuji3* (c) *Flowerfield1* (d) *Flowerfield2* (e) *Farm* (f) *Jetty*

6.9.3 LZC and JPEG2000

In this section we will compare the results of the LZC algorithm with the new JPEG2000 standard to see if the performance of LZC is comparable with JPEG2000. The JPEG2000 test codec is JJ2000, which is an implementation of JPEG2000 standard in $Java^{TM}$ and was downloaded from *http://jj2000.epfl.ch/index.html*. However, unlike SPIHT, the coding algorithm used in JPEG2000 is not related to the zerotree algorithm, so we will just use the comparison as a reference only.

In this comparison we use the most widely used test image, *Lena*, and the cropped JPEG 2000 test, *Bike*. The PSNR results in Figure 6.41 show that LZC can achieve similar objective coding quality to JJ2000 for coding both test images. The results from SPIHT are also plotted for comparison. The compressed *Lena* and *Bike* images are shown in Figure 6.42 and 6.43, respectively. The coding bit-rates are at 0.24 bpp and 0.48 bpp. Despite the objective quality performance of LZC being slightly lower than JJ2000, the compressed *Lena* and *Bike* images show no perceptible quality difference between using LZC or JJ2000 at those two bit-rates. Hence, the LZC algorithm can achieve as good subjective quality performance as the JPEG2000 standard.



(a) (b)

Figure 6.41: PSNR comparison - LZC, JJ2000, and SPIHT (a) *Lena* (b) *Bike*

(a) LZC (b) JJ2000

(c) LZC (d) JJ2000

Figure 6.42: The *Lena* images compressed by (a) LZC at 0.24bpp (b) JJ2000 at 0.24bpp (c) LZC at 0.48bpp (d) JJ2000 at 0.48bpp

(a) LZC (b) JJ2000

(c) LZC (d) JJ2000

Figure 6.43: The *Bike* images compressed by (a) LZC at 0.24bpp (b) JJ2000 at 0.24bpp (c) LZC at 0.48bpp (d) JJ2000 at 0.48bpp

6.10 Summary

In this chapter, we have discussed extensively the proposed LZC algorithm. We also presented two versions of the LZC algorithm which perform recursive and raster tree searches. Based on result comparisons, we conclude that the LZC algorithm not only performs as well as SPIHT, but also provides an alternative low coding memory, low coding complexity implementation of the zerotree algorithm. Hence, the LZC algorithm is more suitable than other zerotree algorithms for hardware implementation. Result comparisons with JPEG2000 have also shown that LZC can achieve the same visual quality as the compression standard.

Chapter 7

3D Listless Zerotree Coding for Color Video

7.1 Introduction

Since the LZC algorithm has good performance for low bit-rate still image coding, it can also be extended for low bit-rate video coding. Therefore, in this chapter we will discuss how we can extend the LZC algorithm from image coding to video coding. For maintaining minimal coding complexity, the wavelet transform is applied to remove both temporal and spatial information correlation. The application of the wavelet transform in both spatial and temporal axes is often called the 3D wavelet transform. Therefore, for convenience, the LZC algorithm for video coding is named 3D Listless Zerotree Coding or 3DLZC for short.

We will firstly discuss the 3D wavelet transform. Then, we will discuss the zerotree symbols, zerotree structure, and significant maps used in 3DLZC. After that, we will discuss the encoding and decoding procedures of 3DLZC. Finally, we will present some simulation results from the 3DLZC test codec.

7.2 3D Wavelet Transform

In the 3DLZC algorithm, as in other video coding schemes, a number of successive video frames are grouped into a group of frames (GOF). Since each GOF only spans a short period of time, normally less than one second, each frame will contain similar or even exactly the same information. Therefore, there will be a significant correlation between successive frames.

As discussed in Section 5.4, temporal correlation can be removed by either motion estimation and compensation (ME/C) or a 3D wavelet transform. Since one of the advantages of the LZC algorithm is low coding complexity, it would be incongruous to implement the computationally expensive ME/C for removing temporal correlation. Therefore, the 3D wavelet transform is a more favorable option for LZC algorithm, due to low computational complexity and high regularity.

A 3D wavelet transform is first performed temporally to remove temporal correlation between frames and then spatially to remove spatial correlation in each temporally transformed frame. For the temporal wavelet transform, the number of frames in a GOF has to be dyadic, due to the down-sampling factor being two. After 1-level temporal wavelet transform, the high temporal frequency subbands will contain the changes between successive frames, due to the motion of objects or camera. The low temporal frequency subbands will contain the unchanged information found in successive frames, such as static background or steady objects. Because of this, the low temporal frequency subbands are still temporally correlated and require further application of temporal wavelet transform for removing this correlation. Temporal wavelet decomposition stops when there is only one low temporal frequency subband in a GOF. This low temporal frequency subband contains the unchanged information of that GOF. On the other hand, the remaining high temporal frequency subbands contain either the changes between successive frames or between temporal subbands.

An ideal situation for temporal wavelet transform is that most of the information in a GOF should be packed into the lowest temporal frequency subband, due to high temporal correlation; while the high temporal frequency subbands will contain little information related to the changes in a GOF, ie. only a small number of coefficients in high temporal frequency subbands. However, such a situation may only occur when coding low motion videos. If there are changing scenes or fast moving objects in a GOF, there will be more information leakage happening in the high temporal frequency subbands [84, 143]. This is similar to the situation when high spatial activities cause a large number of coefficients occurring in high spatial frequency subbands. Hence, this is the reason why we need to further apply the spatial wavelet transform to the temporal frequency subbands.

The application of the spatial wavelet transform to temporal subbands is the same as to still images, and the purpose is to reduce the spatial correlation within each temporal subband for a more compact data representation. Normally low temporal frequency subbands have a large number of coefficients, so they have more spatial correlation; while high temporal frequency subbands have a small number of coefficients, so they have less spatial correlation. Therefore, for achieving better spatial decorrelation, we may need to apply more spatial decomposition levels to low temporal frequency subbands, whereas, only one or two dyadic decomposition

levels may be good enough for high temporal frequency subbands [162, 186]. However, in order to effectively form 3D wavelet coefficient trees, we need to have the same decomposition levels for every temporal subband [168].

Figure 7.1 shows the application of the 3D wavelet transform to a GOF of *Akiyo* sequence. The results of the temporal wavelet transform can be interpreted as follows. Firstly, from the left to the right, the first *H1* subband represents the changes between the first two *Akiyo* frames, the second *H1* subband represents the changes between the second two *Akiyo* frames, and so forth. Secondly, the first *H2* subband represents the changes among the first four *Akiyo* frames, and the second *H2* subband represents the changes among the second four *Akiyo* frames. Thirdly, the *H3* subband represents the changes among all the eight frames in this GOF, while the *L3* subband represents unchanged information in this GOF. From the results we can see that this GOF contains mostly constant information and just a little motion; therefore, after temporal wavelet transform, most of the GOF information is packed into the *L3* subband, and only some motion information leakage occurs in high temporal frequency subbands.

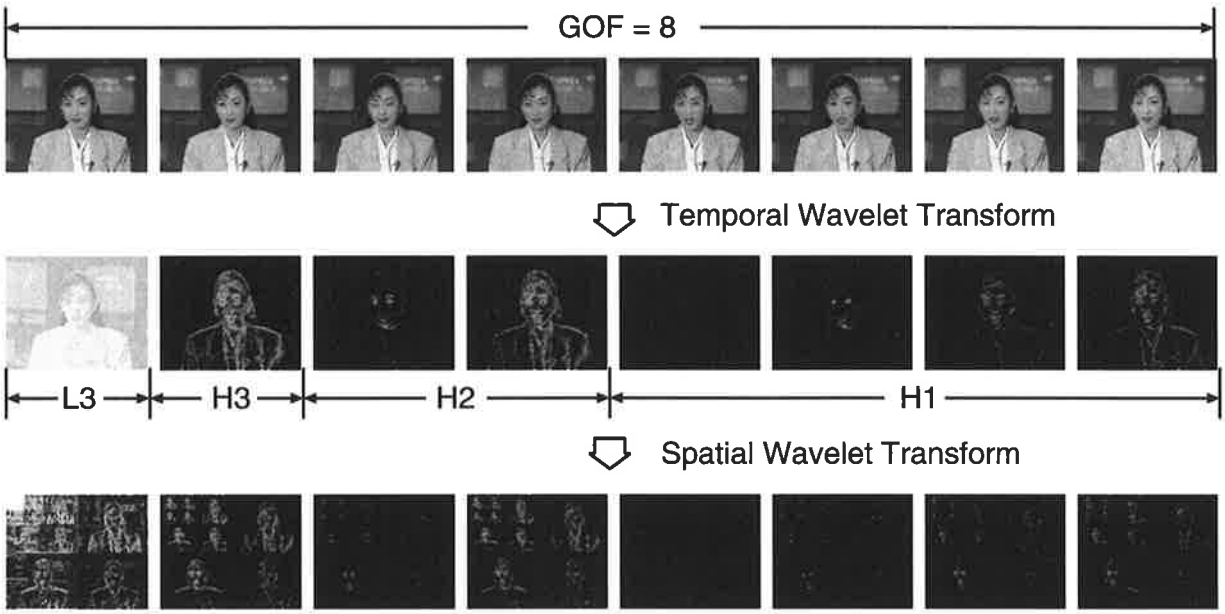


Figure 7.1: The application of 3D wavelet transform to a GOF of *Akiyo* sequence

After applying the spatial wavelet transform to the temporal subbands, the results show that there is not only spatial similarity inside each temporal subband, but also temporal similarity across the temporal subbands. Therefore, this spatiotemporal similarity can be efficiently exploited by the 3DLZC algorithm to form 3D wavelet coefficient trees for insignificant coefficient prediction. That is, the temporal similarity can be used first to predict the insignificant tree across the temporal subbands. However, if there is a lot of motion in a GOF, which causes a large

number of coefficients leaking to high temporal frequency subbands, the spatial similarity can then be exploited to predict insignificant trees inside each temporal subband. Figure 7.2 shows the enlargement of the spatially wavelet decomposed *L3* and *H3* subbands given in Figure 7.1. The spatial similarity between spatial subbands can be used to predict the spatial insignificant trees.

(a) (b)

Figure 7.2: Enlargement of *L3* and *H3* subbands (a) *L3* (b) *H3*

Since the 3DLZC algorithm also performs bit-layer coding for bit-allocation, when implementing the 3D wavelet transform for 3DLZC, we need to make sure that the magnitude dynamic range decreases not only from low spatial frequency subbands to high spatial frequency subbands, but also decrease from low temporal frequency subbands to high temporal frequency subbands. This is because low temporal frequency subbands generally contain more important information and they should be allocated coding bit-budget first.

7.3 3DLZC Zerotree

This section will discuss how the LZC zerotree symbols, zerotree structure, and significant maps should be modified in the 3DLZC algorithm.

7.3.1 3DLZC Zerotree Symbols

After temporal and spatial wavelet decomposition, the wavelet coefficients are classified into three sets: root set R, branch set B, and leaf set L. Coefficients in R are those in the lowest frequency subband and have no parents. Coefficients in L are those in the highest frequency subbands and have no descendants. Coefficients in B are those in neither R nor L and have both parents and descendants.

Suppose the size of a GOF is f frames, and the size of each frame is m rows and n columns, then the classification of wavelet coefficients in a GOF can be formulated as follows.

$$R = \{C(k,i,j) | \text{ for } 0 < k < f/2^{l_t},\ 0 < i < m/2^{l_s} \text{ and } 0 < j < n/2^{l_s}\} \quad (7.1)$$

$$\begin{aligned} B = \{C(k,i,j) | & \text{ for } 0 < k < f/2^{l_t},\ 0 \le i < m/2 \text{ and } 0 \le j < n/2 \\ & \text{except } 0 < i < m/2^{l_s} \text{ and } 0 < j < n/2^{l_s}, \\ & \text{for } f/2^{l_t} \le k < f/2,\ 0 \le i < m/2 \text{ and } 0 \le j < n/2\} \end{aligned} \quad (7.2)$$

$$\begin{aligned} L = \{C(k,i,j) | & \text{ for } 0 < k < f/2,\ 0 \le i < m \text{ and } 0 \le j < n \\ & \text{except } 0 < i < m/2 \text{ and } 0 < j < n/2, \\ & \text{for } f/2 \le k < f,\ 0 \le i < m,\ 0 \le j < n\} \end{aligned} \quad (7.3)$$

In the above formulas, $C(k,i,j)$ represents an individual coefficient, l_t and l_s are respectively the number of temporal and spatial wavelet decomposition levels, and k, i and j are respectively the frame, row and column coordinates.

Figure 7.3 shows an example of the classification of wavelet coefficients into three sets: coefficients in the green region are classified as root set R; coefficients in blue regions are classified as branch set B; coefficients in gray regions are classified as leaf set L. In the figure, the parameter fr, mr, and nr are the frame, row, and column size of the lowest frequency subband and are given by $f/2^{l_t}$, $m/2^{l_s}$, and $n/2^{l_s}$, respectively.

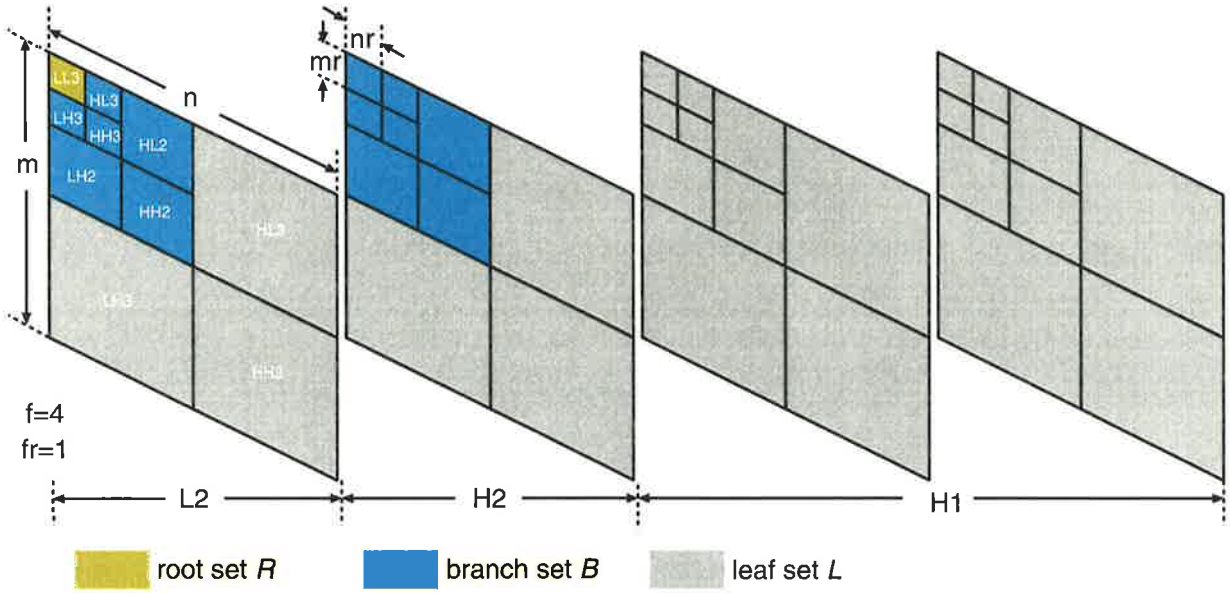


Figure 7.3: Classification of wavelet coefficient sets

The zerotree symbols used in 3DLZC are the same as LZC and are listed as follows:

- $C(k,i,j)$- a wavelet coefficient at coordinate (k,i,j);
- $O(k,i,j)$- the child coefficient set of $C(k,i,j)$;
- $D(k,i,j)$- the descendant coefficient set of $C(k,i,j)$.

There are two parent-child relationships between $C(k,i,j)$ and $O(k,i,j)$. If $C(k,i,j) \in R$, then its children set is $O(k,i,j) = \{(k,i+mr,j),(k,i,j+nr),(k,i+mr,j+nr),(k+fr,i,j),(k+fr,i+mr,j),(k+fr,i,j+nr),(k+fr,i+mr,j+nr)\}$. On the other hand, if $C(k,i,j) \in B$, then its children set is $O(k,i,j) = \{(2k,2i,2j),(2k,2i+1,2j),(2k,2i,2j+1),(2k,2i+1,2j+1),(2k+1,2i,2j),(2k+1,2i+1,2j),(2k+1,2i,2j+1),(2k+1,2i+1,2j+1)\}$.

Like the LZC algorithm, 3DLZC uses symbols C and D to code the positions of wavelet coefficients; while symbol O is used for position reference during encoding and decoding.

7.3.2 3DLZC Zerotree Structure

Like LZC, 3DLZC groups wavelet coefficients into 3D coefficient trees by exploiting the temporal and spatial inter-subband relations. The coefficient grouping enables an effective prediction of a group of insignificant coefficients at the same temporal-spatial locations, as well as an efficient position encoding for those coefficients.

The structure of wavelet coefficient trees is specified by the parent-child relationships discussed in the last section. Figure 7.4 gives an example of the 3DLZC tree structure in a GOF. This GOF has four frames, and has two temporal and three spatial wavelet decomposition levels. The 3D wavelet coefficient tree in the figure has a *root* coefficient in the lowest frequency subband, and the remaining coefficients all descend from this root coefficient. The parent-child relationships of this 3D coefficient tree are only partially indicated by the red arrows. Since the root coefficient is in set R, it has seven children, as linked by the arrows, three in the same temporal subband *L2* and four in the temporal subband *H2*. For convenience, these three in *L2* are named *L-HL*, *L-LH* and *L-HH*, and these four in *H2* are named *H-LL*, *H-HL*, *H-LH* and *H-HH*, representing the roots of tree branches in seven temporal-spatial orientations. On the other hand, coefficients $'a'$ and $'b'$ are in set B, so each of them has eight children. Coefficient $'a'$ has four children in *L2* and four children in *H2*, as indicated by $'A'$; while all eight children of coefficient $'b'$ are in *H1*, as indicated by $'B'$. In this case, there are 255 coefficients descended from the root coefficient. Therefore, if this coefficient tree is insignificant and coded as a zerotree, one root symbol is enough to encode the entire 255 insignificant coefficients, including their positions. This is a very efficient data structure, especially at low coding bit-rates when coefficients are more likely to be below the coding threshold.

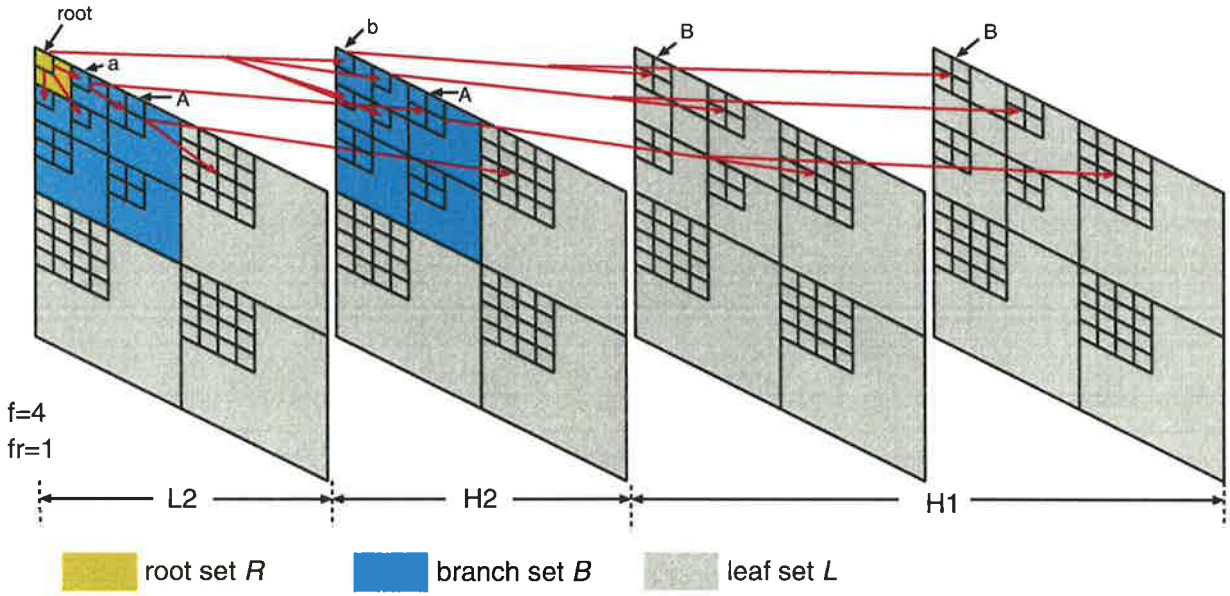


Figure 7.4: 3DLZC wavelet coefficient tree structure

Due to the parent-child relationships of 3DLZC, the dimension of the lowest frequency subband can be either odd or even, while 3D SPIHT requires the lowest frequency subband to have an even dimension, due to its coefficient grouping. Therefore, when coding video with a small frame size, such as 4:1:1 (or 4:2:0) QCIF, the size of the chrominance frames must be adjusted to have the same spatial decomposition levels as the luminance frames. As a result, 3D SPIHT will have more coding complexity than 3DLZC for image size adjustment.

7.3.3 3DLZC Significant Maps

Like LZC, 3DLZC has two types of significant maps: coefficient map F_C used to indicate the significance of wavelet coefficients and descendant map F_D used to indicate the significance of descendant sets.

Figure 7.5 shows an example of how the F_C map and the F_D map can be used to indicate the significance of coefficients and descendant sets in a GOF. Suppose the blue blocks denote significant coefficients, so the corresponding locations in the F_C map are marked by $'1'$; while white blocks denote insignificant coefficients, so the corresponding locations in the F_C map are marked by $'0'$. Similarly, suppose coefficients $'a'$ in *L2* and $'c'$ in *H2* denote the roots of two significant descendant sets $'A'$ and $'C'$, as colored in green, so the corresponding locations in the F_D map are marked by $'1'$. On the other hand, suppose coefficients $'b'$ in *L2* and $'d'$ in *H2* denote the roots of two insignificant descendant sets $'B'$ and $'D'$, as colored in gray, so the corresponding locations in the F_D map are marked by $'0'$.

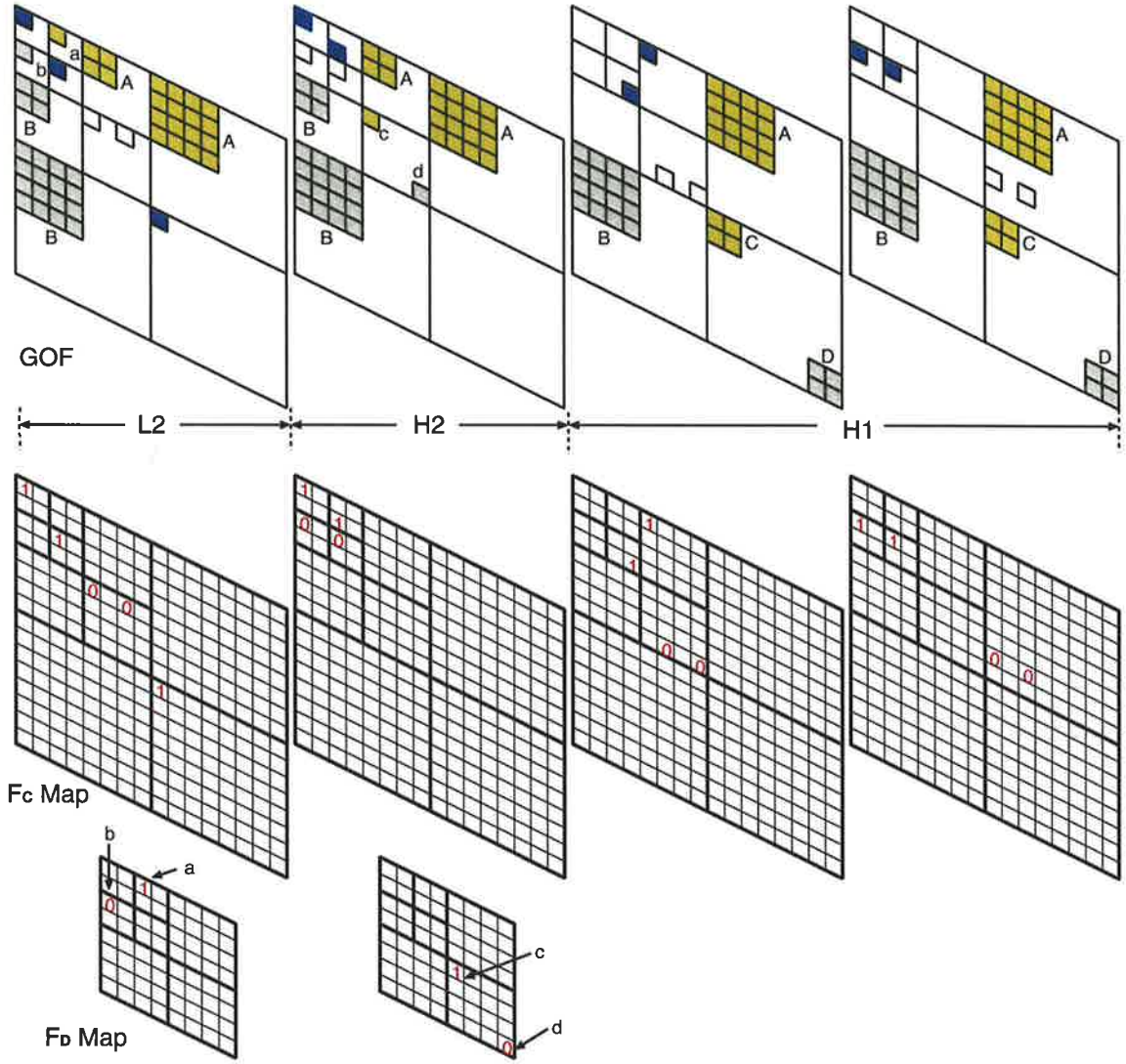


Figure 7.5: 3DLZC significant maps

From Figure 7.5 we can see that the F_C map has the same size as the GOF, while the size of the F_D map is only one eighth of that of GOF. This is because only coefficients in set R and set B have descendants, while those in set L do not have any. Therefore, when coding a color 4:1:1 QCIF sequence with 16 frames in a GOF, the maximum coding memory required by 3DLZC is fixed at 83.53KB [1] for all coding bit-rates. The memory requirement will increase to 334KB for coding color CIF sequences and to 1336KB for coding color 4CIF sequences. On the other hand, the memory required by SPIHT is 1.4MB, as discussed in Section 5.5, when coding QCIF sequence with 16 frames in a GOF. Therefore, the lower memory requirement for maintaining two significant maps makes 3DLZC a much more friendly video coding algorithm than 3D SPIHT for hardware implementation.

[1] Map F_C: [176x144 + 88x72x2(colors)] x 16(GOF) = 608256 bits = 74.25 KB
Map F_D: [176x144 + 88x72x2(colors)] /8 x 16(GOF) = 76032 bits = 9.28 KB

7.4 The 3DLZC Algorithm

In this section we will discuss how the LZC algorithm is extended to 3DLZC for coding video sequences. The discussions in this section include the encoding procedures and decoding procedures of the 3DLZC algorithm. Although there are two versions of LZC algorithms that perform recursive and raster tree search, we shall only preset the recursive tree search 3DLZC algorithm in this section. The process of extending raster tree search LZC to 3DLZC will be similar to that discussed in this section.

7.4.1 3D Recursive Tree Search

The 3DLZC codec always begins the recursive tree search from the root coefficients in set R, and searches all of the descendant coefficients in a temporal and spatial tree-depth fashion. The coordinates of a root coefficient are used to derive the locations of its descendants in the 3D wavelet matrix by the 3DLZC parent-child relationships. Figure 7.6 illustrates the order of the 3D recursive tree search in a GOF, which is two level temporal and two level spatial wavelet decomposed. The number marked on each coefficient indicates the order of visitation by the 3DLZC codec.

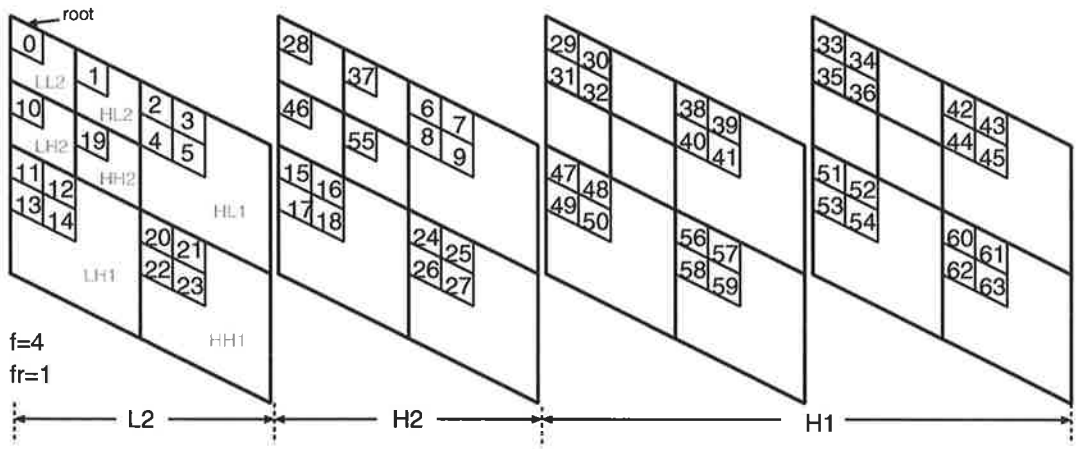


Figure 7.6: The 3D recursive tree search order of 3DLZC

Figure 7.7 shows an example of how the 3D recursive tree search covers all of the coefficients in a GOF. There are four temporal-spatial wavelet trees, which are colored differently. The 3DLZC codec first searches the tree with green color from its root coefficient. After that, the 3DLZC codec will move to the next tree root in the raster position, as indicated by the red arrows. Therefore, after a raster search of the tree roots, the 3DLZC codec will have searched all of the coefficients in a GOF.

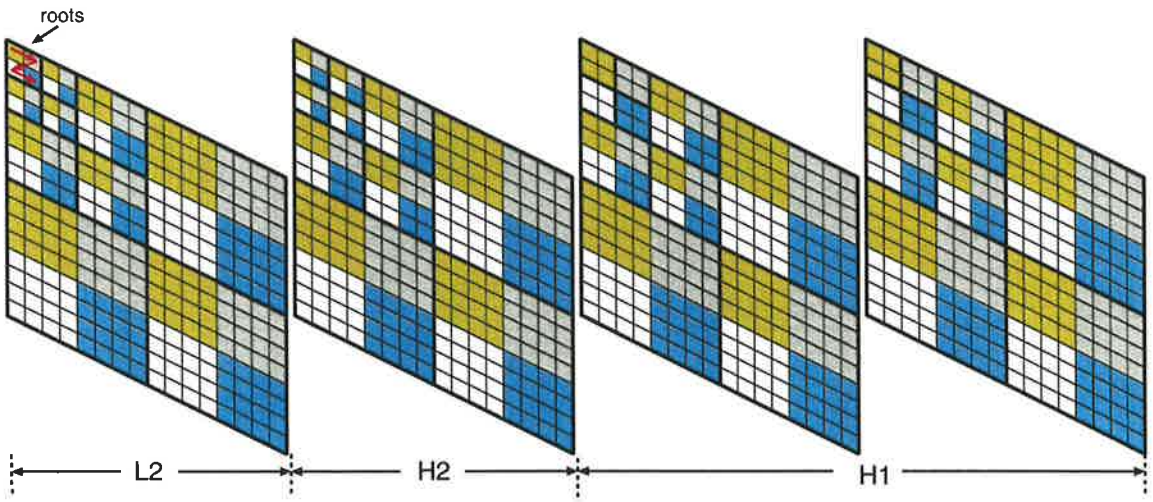


Figure 7.7: The 3D recursive tree search for a GOF

7.4.2 Encoding Procedures

Based on extension of the LZC algorithm, the 3DLZC algorithm also has two encoding procedures: the main encoding procedure and the tree encoding procedure. The main encoding procedure mainly controls the coding flow, including the choice of coding threshold, the color coding order, and the starting and terminating point in the 3D matrix. The tree encoding procedure mainly encodes individual wavelet coefficients and partitions wavelet coefficient trees in the 3D matrix. The tree encoding procedure is called by itself and by the main encoding procedure.

The main encoding procedure of the 3DLZC algorithm is presented in Algorithm 7.1. The video sequences are assumed to be in YUV color space, but may also be in YCbCr color space. The procedure consists of four main steps.

In Step 1, the 3DLZC codec clears the F_C and F_D maps for YUV GOFs, and outputs a number n in bits, which is the the word-length of the largest coefficient among the YUV GOFs. The number n gives the initial coding threshold as 2^n.

In Step 2, the 3DLZC codec encodes the root coefficients in set R and follows the color coding order Y→V→U. Suppose a root coefficient $C(k,i,j)$ is coded. The 3DLZC codec will first check its significance in the F_C map. If it is significant, ie $F_C(k,i,j) = 1$, then the codec will output the nth significant bit of $C(k,i,j)$. On the other hand, if $C(k,i,j)$ is insignificant, ie $F_C(k,i,j) = 0$, then the codec will test its significance against the current threshold by using function $S_n()$. If the test result is significant, then the codec will output the sign and mark the corresponding location in the F_C map as '1'. Otherwise, the codec will move to the next root

coefficient at the raster location. The root coefficient encoding process finishes when all of the roots in the YUV GOFs have been visited by the codec once.

In Step 3, the 3DLZC codec encodes the descendant sets (or 3D wavelet trees) of the root coefficients. Suppose a descendant set $D(k,i,j)$ is encoded. The codec will first check its significance in the F_D map. If $D(k,i,j)$ has been found to be significant, ie. $F_D(k,i,j) = 1$, then the codec will call the tree encoding procedure to perform 3D tree branch encoding. On the other hand, if $D(k,i,j)$ has previously been encoded as a zerotree, ie. $F_D(k,i,j) = 0$, then the codec will test its significance against the current threshold by using function $S_n()$. If the test result is significant, then the codec will mark the corresponding root location in the F_D map as $'1'$ and call the tree encoding procedure for 3D tree branch encoding. Otherwise, the codec will stop encoding at the current root coefficient and move to the next root coefficient at the raster location. The 3D wavelet tree encoding finishes when all of the trees rooted in set R have been searched once. Since each root in set R has seven children, representing the roots of tree branches in seven temporal-spatial orientations, the codec calls the tree encoding procedure seven times for each color component.

In Step 4, the 3DLZC codec decreases the number n by one for encoding the next bit-layer and repeats the same encoding process from Step 2, again.

The tree encoding procedure is presented in Algorithm 7.2. The 3DLZC codec first encodes an individual coefficient and then the tree branch rooted at that coefficient. The encoding processes on the coefficient and the 3D tree branch are similar to Step 2 and Step 3 in the main encoding procedure, respectively. However, the tree branch encoding is performed by recursively calling the tree encoding procedure itself. Since a coefficient in set B has eight children, the tree encoding procedure needs to call itself eight times. The recursive procedure call is for partitioning significant tree branches, but it stops when the coordinates received are in set L, because coefficients in set L do not have child coefficients.

Figure 7.8 shows the color coding order of 3DLZC. Since wavelet coefficients in set R are the most significant and contain the most GOF information, the 3DLZC codec allocates bit-budget to them by following the same color coding order as LZC, ie. Y→V→U. After that, the 3DLZC codec encodes coefficients in B sets and L sets of YUV trees. Within each tree, the encoding order for tree branches is *L-HL* → *L-LH* → *L-HH* → *H-LL* → *H-HL* → *H-LH* → *H-HH*. When finished allocating bit-budget to the coefficients and trees on the current bit-layer, the 3DLZC codec decreases n by one and repeats the same bit-allocation order on the next bit-layer.

After encoding several bit-layers, the bit-stream generated by the 3DLZC codec will look like the one shown in Figure 7.9. Like LZC, the 3DLZC bit-stream also has the information ordering property, ie. embeddedness. Therefore, the bit-stream can be truncated at any point to

```
Procedure MainEncoding
Step 1:
  clear F_C and F_D;
  for all Y, U, and V GOF
   output n = ⌊log2(max_(k,i,j)|C(k,i,j)|)⌋;
  clear F_C and F_D;
Step 2:
  for all Y, U, and V GOF
   for all C(k,i,j) ∈ SET R do
     if F_C(k,i,j) = 1 then
       output the nth significant bit of |C(k,i,j)|;
     else
       output S_n(C(k,i,j));
       if S_n(C(k,i,j)) = 1 then
         output the sign of C(k,i,j)
         and set F_C(k,i,j) = 1;
Step 3:
  for all Y, U, and V GOF
   for all D(k,i,j) ∈ SET R do
     if F_D(k,i,j) = 1 then
       for each C(z,x,y) ∈ O(k,i,j)
       → EncodeTree(z,x,y,n);
     else
       output S_n(D(k,i,j));
       if S_n(D(k,i,j)) = 1 then
         set F_D(k,i,j) = 1;
           for each C(z,x,y) ∈ O(k,i,j)
           → EncodeTree(z,x,y,n);
Step 4:
  decrease n by 1 and go back to Step 2;
End
```

Algorithm 7.1: 3DLZC Main Encoding Procedure

```
Procedure EncodeTree(k,i,j,n)
 if F_C(k,i,j) = 1 then
    output the nth significant bit
    of |C(k,i,j)|;
 else
    output S_n(C(k,i,j));
    if S_n(C(k,i,j)) = 1 then
       output the sign of C(k,i,j) and
       set F_C(k,i,j) = 1;

 if C(k,i,j) NOT IN SET L
    if F_D(k,i,j) = 1 then
       for each C(z,x,y) ∈ O(k,i,j)
        →EncodeTree(z,x,y,n);
    else
       output S_n(D(k,i,j));
       if S_n(D(k,i,j)) = 1 then
          set F_D(k,i,j) = 1;
          for each C(z,x,y) ∈ O(k,i,j)
          →EncodeTree(z,x,y,n);
End.
```

Algorithm 7.2: 3DLZC Tree Encoding Procedure

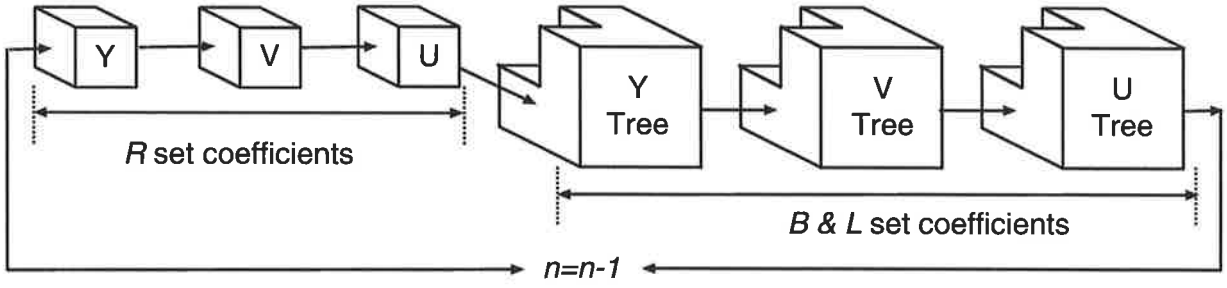


Figure 7.8: Example of 3DLZC encoding order on wavelet decomposed YUV GOFs

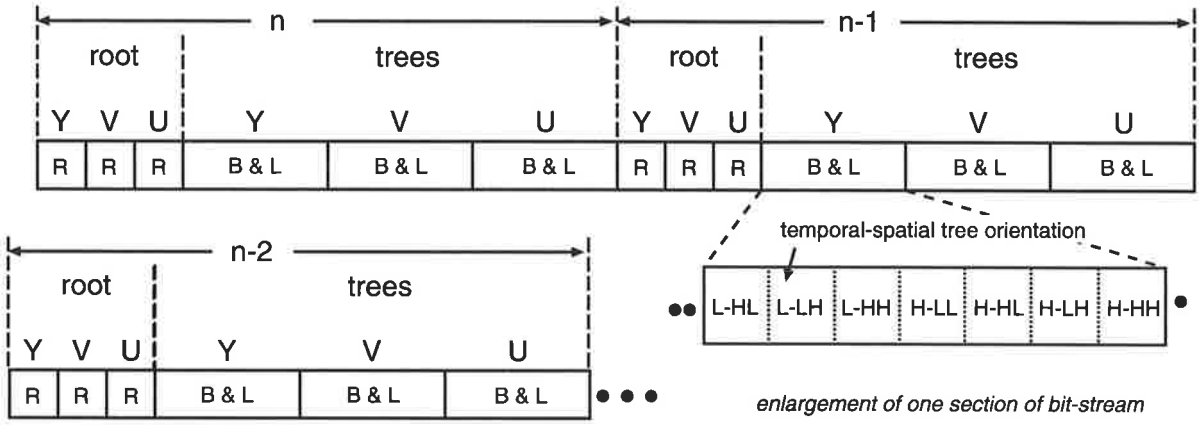


Figure 7.9: Example of 3DLZC video bit-stream after encoding several bit-layers

give a very precise bit-rate or quality control.

7.4.3 Decoding Procedures

Since 3DLZC is a unitary coding algorithm, it also has two decoding procedures: the main decoding procedure presented in Algorithm 7.3 and the tree decoding procedure presented in Algorithm 7.4. Both decoding procedures follow exactly the same coding process as their encoding counterparts, so they are simply presented here without further discussion. The only differences are that the 'output' process in encoding becomes the 'input' process in decoding, and the coefficient magnitude test in encoding becomes magnitude reconstruction in decoding. The process of reconstructing wavelet coefficients for 3DLZC is the same as for LZC, so readers are referred back to Section 6.4.3 for more details.

```
Procedure MainDecoding
Step 1:
  input n;
  clear F_C and F_D;
Step 2:
  for all Y, U, and V GOF
   for all C(k,i,j) ∈ SET R do
     if F_C(k,i,j) = 1 then
       input the nth significant bit of |C(k,i,j)| and
       refine C(k,i,j);
     else
       input S_n(C(k,i,j));
       if S_n(C(k,i,j)) = 1 then
         input the sign of C(k,i,j) and
         reconstruct C(k,i,j) = (sign) 1.5 × 2^n;
         and set F_C(k,i,j) = 1;
Step 3:
  for all Y, U, and V GOFs
   for all D(k,i,j) ∈ SET R do
     if F_D(k,i,j) = 1 then
       for each C(z,x,y) ∈ O(k,i,j)
       → DecodeTree(z,x,y,n);
     else
       input S_n(D(k,i,j));
       if S_n(D(k,i,j)) = 1 then
         set F_D(k,i,j) = 1;
           for each C(z,x,y) ∈ O(k,i,j)
           → DecodeTree(z,x,y,n);

Step 4:
  decrease n by 1 and go back to Step 2;
End
```

Algorithm 7.3: 3DLZC Main Decoding Procedure

```
Procedure DecodeTree(k,i,j,n)
 if F_C(k,i,j) = 1 then
    input the nth significant bit of |C(k,i,j)|
    and refine C(k,i,j);
 else
    input S_n(C(k,i,j));
    if S_n(C(k,i,j)) = 1 then
       input the sign of C(k,i,j) and
       reconstruct C(k,i,j) = (sign) 1.5 × 2^n;
       set F_C(k,i,j) = 1;

 if C(k,i,j) NOT IN SET L
    if F_D(k,i,j) = 1 then
       for each C(z,x,y) ∈ O(k,i,j)
        →DecodeTree(z,x,y,n);
    else
       input S_n(D(k,i,j));
       if S_n(D(k,i,j)) = 1 then
          set F_D(k,i,j) = 1;
          for each C(z,x,y) ∈ O(k,i,j)
          →DecodeTree(z,x,y,n);
End.
```

Algorithm 7.4: 3DLZC Tree Decoding Procedure

7.5 3DLZC Test Codec and Test Video Sequences

This section will briefly discuss the 3DLZC test codec and the test video sequences.

7.5.1 The 3DLZC Test Codec

The test codec used to verify the 3DLZC algorithm was written in ANSI C language and consists of four modules for both encoding and decoding. The functions of those four modules are color space transform, 3D wavelet transform, 3DLZC and arithmetic coding, as shown in Figure 7.10. Before coding, a video sequence is always separated into a series of GOF, and each GOF has a dyadic number of video frames. After color space transform, the chrominance color components can be further down-sampled to reduce the data size. Therefore, in decoding, the 3DLZC test codec may need to up-sample those down-sampled chrominance color components before color space transform.

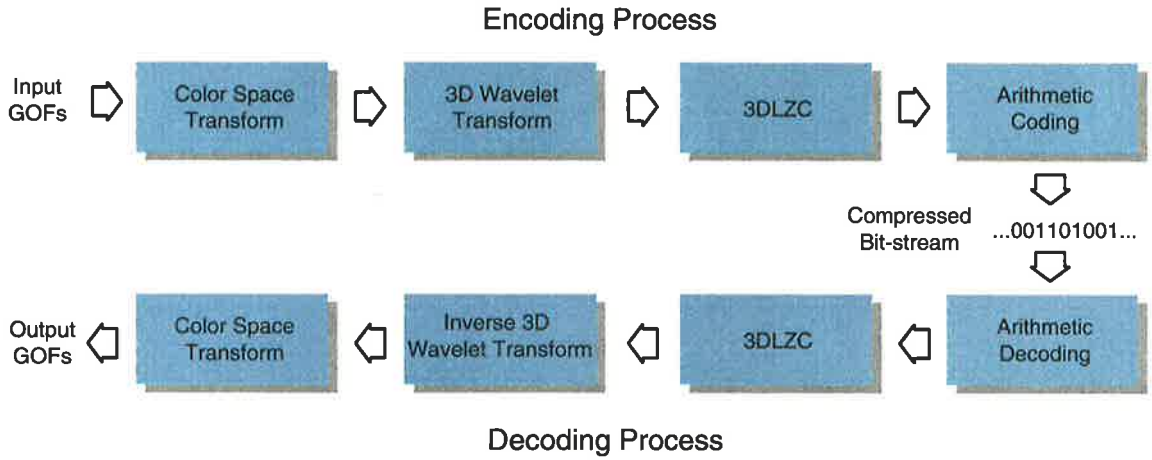


Figure 7.10: Modules of the 3DLZC test codec

7.5.2 Test Video Sequences

Figure 7.11 shows the commonly used test video sequences: *Akiyo*, *Hall Monitor*, *Carphone*, *Mother-daughter*, *Silent*, and *Container*. They are all in QCIF format (176×144) and are stored in YUV color space. Also, the chrominance sequences of those test video sequences are horizontally and vertically down-sampled by a factor of two, so the size of each chrominance frame is only 88×72. This file format is known as 4:1:1 (or 4:2:0). The frame rate for those test sequences is 30 fps (frame per second).

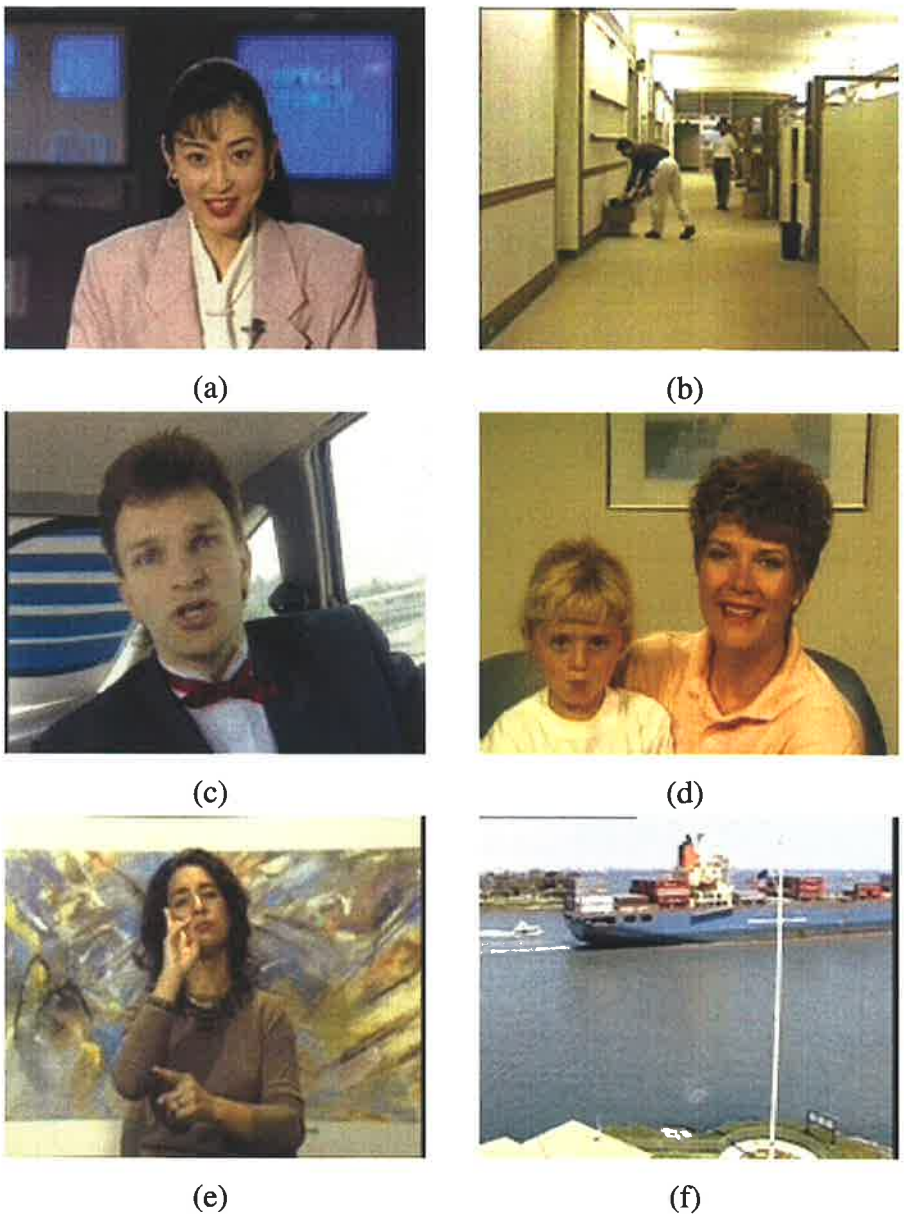


Figure 7.11: QCIF video sequence (a) *Akiyo* (b) *Hall Monitor* (b) *Carphone* (c) *Mother-daughter* (e) *Silent* (f) *Container*

7.6 Preliminary Experiment

As mentioned in the previous section, the video sequence is separated into a series of GOFs for encoding, with a dyadic number of frames in a GOF, 4 frames, 8 frames, 16 frames or 32 frames. The choice of GOF size has a direct influence on the performance of temporal decorrelation, which in turn influences the 3DLZC coding performance. In this section, we will examine how the choice of a GOF size affects the performance of the 3DLZC test codec.

The presumption in using the temporal wavelet transform is that most of the information will be packed into the lowest temporal frequency subband, while high temporal frequency subbands only contain changes within the GOF. Therefore, the higher the number of frames in a GOF, the more efficient will be the information decorrelation and compaction by temporal wavelet transform. However, the higher the number of frames in a GOF, the longer will be the time span for that GOF. As a result, the GOF may contain too much motion information, causing high temporal frequency subbands to contain too many coefficients that represent changes in the GOF. Hence, the choice of 32 frames is a GOF is not appropriate, as the GOF will span around 3 seconds for 10 fps and will contain too much motion information.

However, if the the number of frames in a GOF is too small, temporal correlation may not be removed effectively. In other words, a small GOF may result in good information decorrelation and compaction within that GOF, but between successive GOFs there may be a significant amount of information correlation as well. This inter-GOF correlation may actually need to be encoded with more coding bit-budget. Hence, the choice of 4 frames in a GOF is not efficient, because the GOF will only span around 0.4 second for 10 fps and the successive GOFs may contain similar or even the same information. Furthermore, 4 frames in a GOF is too short for data extension when performing the temporal wavelet transform, while 32 frames in a GOF is too long and will increase the coding latency.

Consequently, the choice of GOF size is only between 8 or 16 frames. Figure 7.12 shows the experimental results from coding *Akiyo*, *Hall Monitor*, and *Carphone* sequences at 60000 bps (bits per second) with the frame rate of 10 fps by using 8 or 16 frames for a GOF. The options for the 3DLZC test codec were set as follows. The filter sets used for temporal and spatial wavelet transform were respectively the Haar filter set and the 9/7 filter set [89]. This is because the Haar filter set is commonly used for temporal wavelet transform [161, 168, 186] and gives the least implementation complexity, and the 9/7 filter gives a good performance for spatial wavelet decomposition as discussed in Section 6.6.1 and Section 6.8.2. The number of decomposition levels was three for both temporal and spatial wavelet transforms. The signal extension method used was symmetric extension.

The results show that when coding slow motion and steady background sequences, such

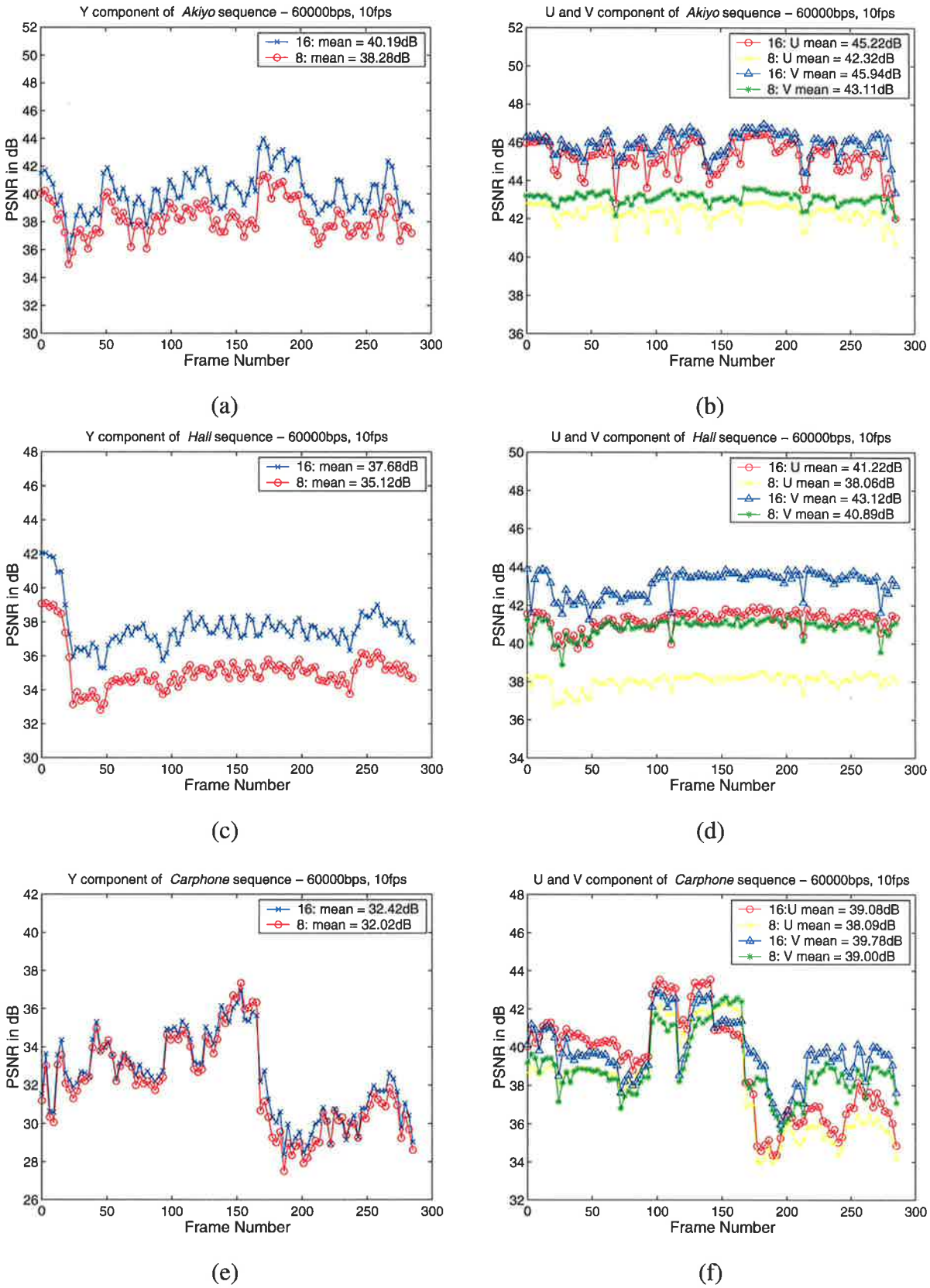


Figure 7.12: PSNR comparison for different GOF sizes (a) *Akiyo* Y component (b) *Akiyo* U & V components (b) *Hall Monitor* Y component (c) *Hall Monitor* U & V components (e) *Carphone* Y component (f) *Carphone* U & V components

as *Akiyo* and *Hall Monitor*, using 16 frames for a GOF gives better coding performance for the 3DLZC test codec. For coding high motion and changing background sequences, such as *Carphone*, using 16 frames for a GOF only gives a slight improvement in coding performance over using 8 frames. Therefore, based on the results, we may want to use 16 frames for a GOF in 3DLZC, as it generally gives a better performance for coding both low motion and high motion video sequences. This also coincides with the findings of Kim et al. [168]

7.7 Results

In this section we will firstly compare the performance of the 3DLZC algorithm with the H.263+ standard [187] for verifying the use of the temporal wavelet transform for temporal information decorrelation. After that, we will briefly compare the performance of 3DLZC with the published 3D SPIHT results.

7.7.1 3DLZC and H.263

Since the motivation for deriving the 3DLZC algorithm is for low bit-rate video coding with low coding complexity, the temporal wavelet transform is exploited instead of the computationally expensive ME/C scheme. Therefore, in this section we will compare the performance of 3DLZC with the low bit-rate video compression standard H.263+, based on DCT and ME/C, to verify the performance difference between using the temporal wavelet transform and ME/C.

The coding options for the 3DLZC test codec were set to 16 frames for a GOF, Haar filter set for temporal wavelet transform and 9/7 filter set for spatial wavelet transform, 3 level temporal and spatial wavelet decomposition, and symmetric signal extension method. The test H.263+ codec used was tmn8 which was downloaded from ftp://dspftp.ece.ubc.ca/pub/tmn [2]. All of tmn8's advanced options (-D, -E, -F) were enabled. The QCIF test video sequences used for the simulation were *Akiyo*, *Hall Monitor*, *Carphone*, *Mother-daughter*, *Silent*, and *Container*, as shown in Figure 7.11. They can be classified into four categories: *Akiyo* and *Mother-daughter* are head-and-shoulder sequences with moderate motion and steady background; *Hall Monitor* and *Container* are monitoring sequences with slow moving objects; *Carphone* is video-phone sequence with rapid scene changes outside the car window; *Silent* is a sign language sequence with rapid hand movements. Those sequences were coded at 60000 bps with 10 fps, ie. coding every third frame in the test sequence, and each frame was allocated 6000 bits. Therefore, each 16-frame GOF was allocated a total bit-budget of 96000 bits which were distributed to the YUV GOFs by the 3DLZC coding procedures discussed in Section 7.4.2.

[2]However, tmn8 is no longer publicly available. Please refer to http://spmg.ece.ubc.ca/h263plus/h263plus.html for more detail.

Although we have argued that PSNR does not predict well the perceived visual quality, it is not feasible to show a whole video sequence in print for subjective quality comparison. Therefore, we have chosen to present the results from frame by frame PSNR comparison for the whole video sequence, because PSNR is the simplest metric for quality evaluation as well as the most commonly used result comparison method in the video coding literature [158, 161, 162, 165, 168, 186]. The PSNR comparisons for all of the test video sequences are presented in Figure 7.13 and 7.14, and the comparisons are separated into luminance sequence, Y, and chrominance sequences, U and V. The results shown are from frame 0 to frame 287.

From the comparison of Y sequences we can see that the PSNR values of 3DLZC coded sequences have similar envelopes as those of H.263+ coded sequences, despite the 3DLZC coded sequences having lower mean PSNR values than the H.263+ coded sequences. This result verifies that the temporal wavelet transform is able to achieve similar performance on temporal decorrelation as the ME/C scheme, but with much lower coding complexity. The 3DLZC even has a very similar coding performance with H.263+ when coding fixed-background, slow moving object sequences, such as *Hall Monitor* and *Container*. This suggests that the ME/C can be replaced by the temporal wavelet transform under such coding conditions. However, similarly to other 3D wavelet video coding algorithms, we observe the PSNR dip at the boundary of each GOF from those sequences coded by 3DLZC. This is partly due to object motion and partly due to extension for the temporal wavelet transform [168].

The PSNR comparisons of U and V sequences show that 3DLZC performs better than H.263+ in coding chrominance sequences, regardless of high motion or low motion video sequences. This suggests that chrominance sequences contain more low temporal frequency information but less high temporal frequency information, ie. higher temporal correlations. Therefore, chrominance sequences can be more effectively decorrelated by the temporal wavelet transform, due to its high information compaction property.

Figures 7.15 and 7.16 show the reconstructed video frames of the compressed video sequences. The original versions of those video frames are shown in Figure 7.11. These frames were chosen because they are also presented in the 3D SPIHT literature [168]. From the visual comparisons of those sample frames we can see that for most cases there is not much visual quality difference between 3DLZC or H.263+ coded frames. However, for *Silent* sequence, the frame reconstructed from 3DLZC has lost some hand detail.

In short, 3DLZC has shown a comparable performance in terms of PSNR values and reconstructed visual quality. In particular, for coding low motion video sequences, 3DLZC achieves similar coding performance to H.263+, but with much lower coding complexity which is the result of performing the regular 3D wavelet transform instead of the irregular ME/C scheme for data decorrelation. Also, for coding chrominance sequences, 3DLZC outperforms H.263+.

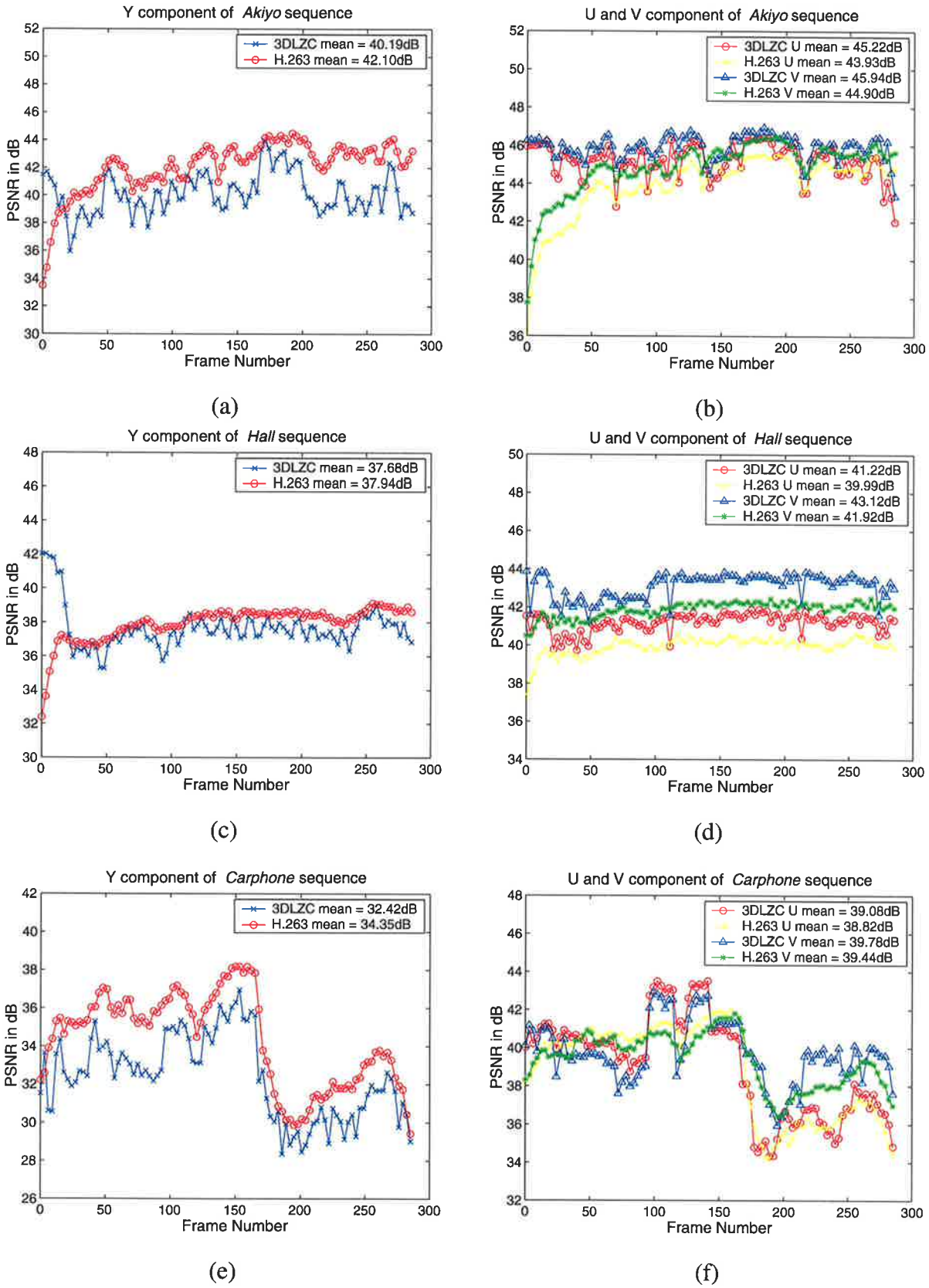


Figure 7.13: PSNR comparison - 3DLZC & H.263 (a) *Akiyo* Y component (b) *Akiyo* U & V components (c) *Hall Monitor* Y component (d) *Hall Monitor* U & V components (e) *Carphone* Y component (f) *Carphone* U & V components

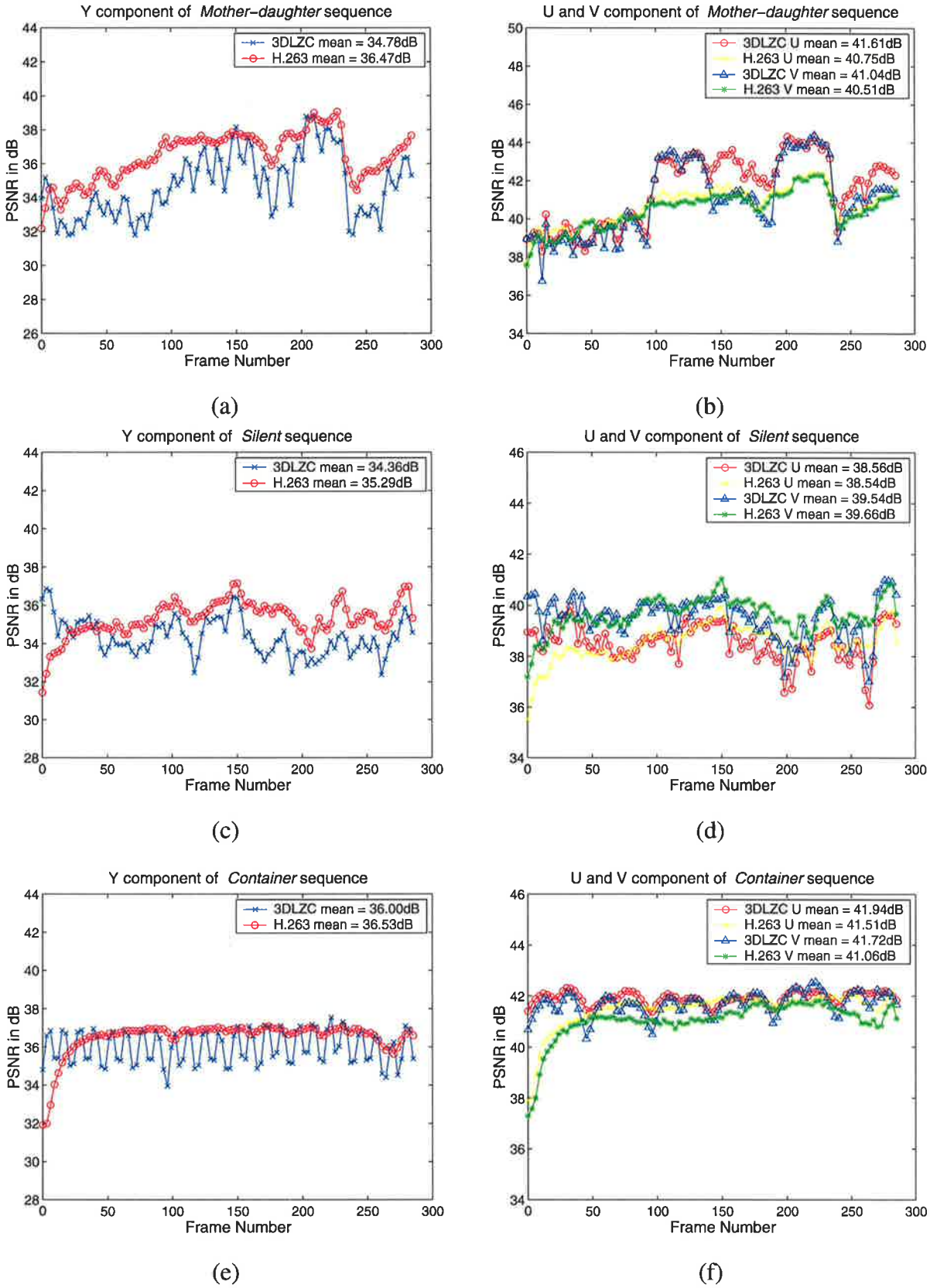


Figure 7.14: PSNR comparison - 3DLZC & H.263 (a) *Mother-daughter* Y component (b) *Mother-daughter* U & V components (c) *Silent* Y component (d) *Silent* U & V components (e) *Container* Y component (f) *Container* U & V components

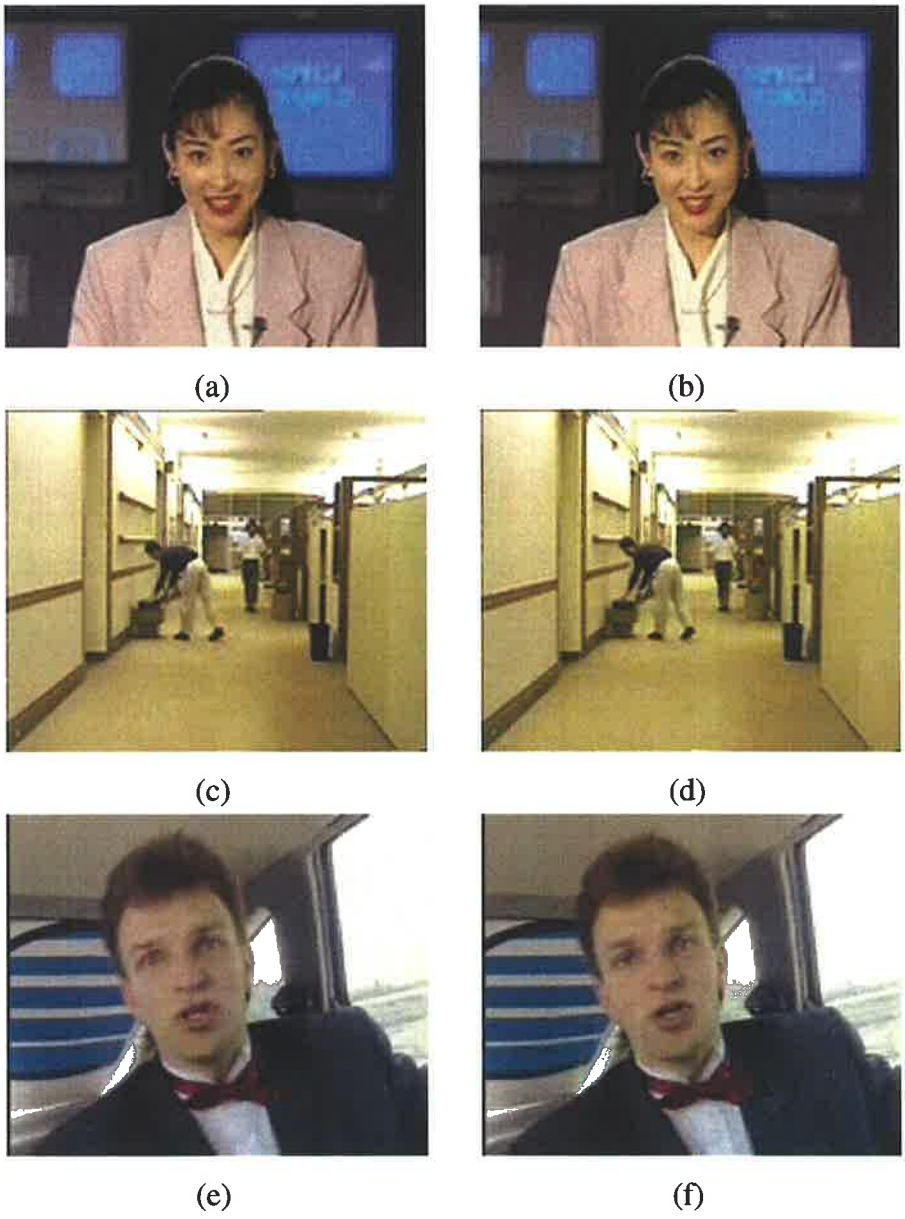


Figure 7.15: Visual quality comparison of compressed video frames - 3DLZC & H.263: *Akiyo* frame 228 (a) 3DLZC (b) H.263; *Hall Monitor* frame 123 (c) 3DLZC (d) H.263; *Carphone* frame 150 (e) 3DLZC (f) H.263

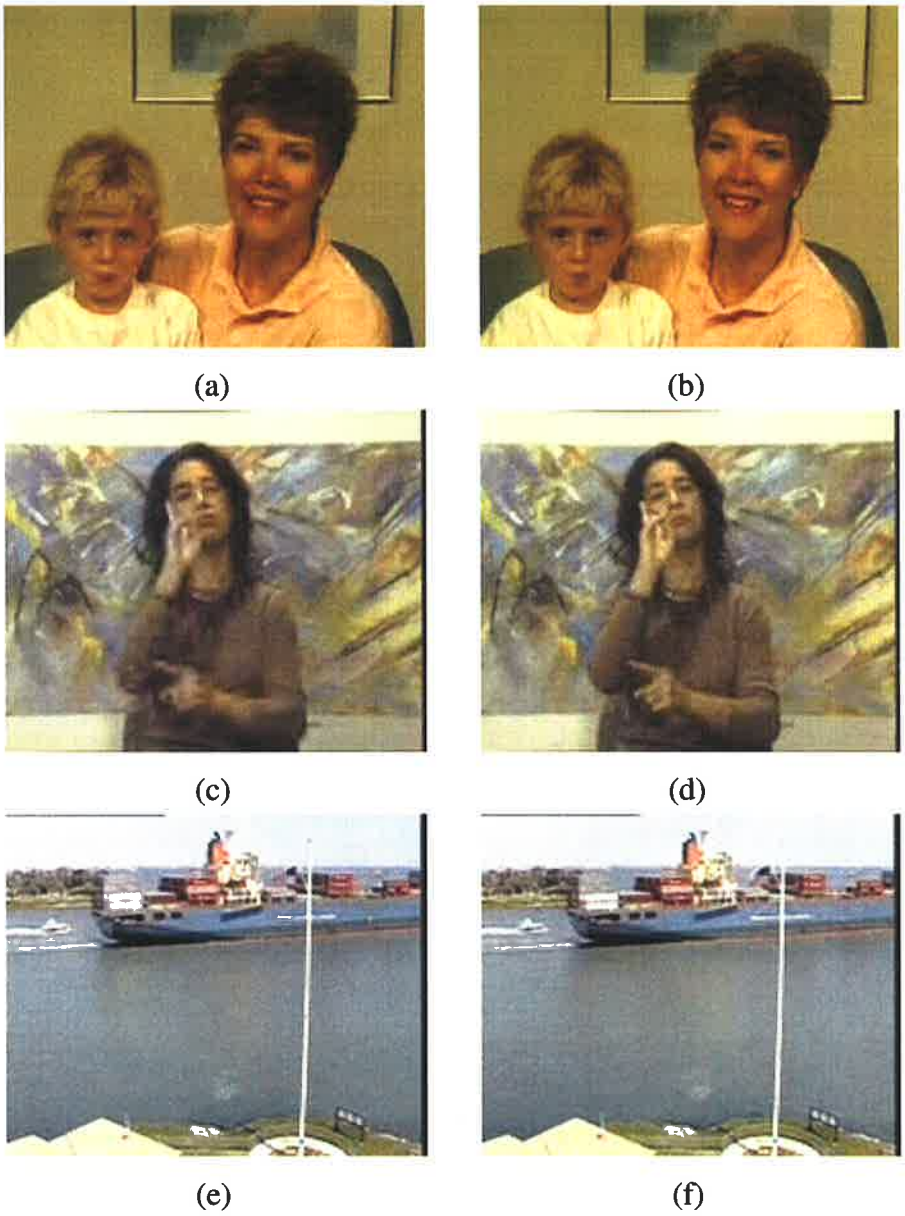


Figure 7.16: Visual quality comparison of compressed video frames - 3DLZC & H.263: *Mother-daughter* frame 102 (a) 3DLZC (b) H.263; *Silent* frame 243 (c) 3DLZC (d) H.263; *Container* frame 048 (e) 3DLZC (f) H.263

7.7.2 3DLZC and 3D SPIHT

In this section we briefly compare 3DLZC results with the 3D SPIHT results published in [168]. The coding options for the 3DLZC test codec were set to be the same as mentioned in the previous section. Table 7.1 shows the mean PSNR comparison of 3DLZC and 3D SPIHT for the *Hall Monitor* sequence, the only average PSNR available in [168]. The coding bit-rates are 60000 bps and 30000 bps, and the frame rate is 10 fps.

At 60000 bps, 3D SPIHT is 0.27 dB better than 3DLZC in coding Y sequence. However, 3DLZC is 0.81 dB and 0.74 better than 3D SPIHT in coding U and V sequence, respectively. Furthermore, at 30000 bps, 3DLZC is 0.39 dB better than 3D SPIHT in coding Y sequence, but 3DLZC is 0.31 dB worse and only 0.09 dB better than 3D SPIHT in coding U and V sequence, respectively. Alternatively, readers can also compare the 3DLZC results in Figure 7.13 and Figure 7.14 with those presented in [168] for the quality comparison of the whole video frames.

bps	Y		U		V	
	3DLZC	3D SPIHT	3DLZC	3D SPIHT8	3DLZC	3D SPIHT
60000	37.68	37.95	41.22	40.41	43.12	42.38
30000	33.34	32.95	37.84	38.15	40.77	40.68

Table 7.1: Mean PSNR comparison of *Hall Monitor* sequence - 3DLZC & SPIHT

Although we may not have a through performance comparison by using just one video sequence here, the comparison has shown 3DLZC can perform as well as or even better than 3D SPIHT, but with a lot less coding memory requirement.

7.8 Summary

In this chapter we have presented the 3DLZC algorithm for video coding as an extension of the LZC algorithm. Inherited from LZC, 3DLZC requires no coding lists, which results in a low memory zerotree video coding algorithm. 3DLZC also has a very precise bit-rate control, due to its progressive bit-layer coding. Furthermore, in the attempt to minimize the coding complexity, 3DLZC exploits wavelet transforms for both temporal and spatial information decorrelation. The result comparisons have shown that 3DLZC has a comparable performance to H.263+ for low bit-rate video coding, especially for coding low motion video sequences. In addition, the comparison between 3DLZC and 3D SPIHT has suggested that 3DLZC can perform as well as or better than 3D SPIHT, but with a much lower coding memory requirement.

Chapter 8

Possible Extensions to the LZC algorithm

8.1 Introduction

This chapter suggests two possible extensions for the LZC algorithm. One is to implement a visual frequency weighting adjustment to the LZC codec for better compression results. The other is to implement a video layer separation function to the 3DLZC codec for adding some functionality.

At the time of writing this thesis, the frequency weighting adjustment feature was successfully added to the LZC codec. However, due to time limitations, the video layer separation function was not able to be added to the 3DLZC codec. Nevertheless, a simple video layer separation algorithm is proposed in this chapter, leaving the possibility for implementing this function to future work.

8.2 Visual Frequency Weighting Adjustment for LZC

This section investigates the effects of implementing frequency weighting adjustment to the LZC codec for improving bit-allocation efficiency and compression results.

8.2.1 Visual Frequency Weighting

As we have mentioned in Section 6.2, the bit-allocation scheme for the LZC algorithm follows the order of visual information importance. That is, the LZC codec always allocates coding bit-budget to the visually more important wavelet subbands before others. In the proposed

LZC algorithm, the way to ensure that the bit-allocation sequence follows the order of visual information importance is to have the wavelet subbands decrease in magnitude with respect to their corresponding CSFs. This is achieved by taking advantage of the wavelet filter coefficient, $\sqrt{2}$, so that the magnitudes of wavelet subbands decrease from the lowest frequency subband *LL* to the highest frequency subband *HH*.

However, this method of mapping wavelet subbands to the CSFs to reflect their visual importance is efficient, but not yet close to optimal. A better way is to have one frequency weight per subband, with each frequency weight determined from the CSF measurement of that subband [144, 178]. That is, if a subband has a higher CSF, it is perceptually more important and will have a higher frequency weight. On the other hand, if a subband has a smaller CSF, it will have a smaller frequency weight, as the loss of information in that subband will not incur noticeable loss of visual quality. Since the frequency weights are real numbers, the magnitude adjustment of each wavelet subband can more precisely reflect the corresponding CSF and therefore the visual importance. Hence, after frequency weighting adjustment, the LZC codec can more accurately allocate bit-budget to encode those more perceptually important wavelet coefficients to give better compression results.

One example of the visual frequency weighting set for YCbCr color space are shown in Table 8.1. This set of frequency weightings are recommended in JPEG2000 for a viewing distance of 1700 pixels [178]. For other viewing distances there will be different frequency weighting sets.

8.2.2 Results

This extra frequency weighting adjustment was added to the LZC test codec as an additional feature, and the frequency weights used were those shown in Table 8.1. The codec was tested on those images shown in Appendix A, but we only show the results from compressing the *Girls* image, as it shows the potential benefits of frequency weighting adjustment more clearly. Figure 8.1(a) shows part of the reconstructed *Girls* images with frequency weighting adjustment, and Figure 8.1(b) shows part of the reconstructed *Girls* images without frequency weighting adjustment. Both images were compressed at 1.5 bpp (or 16:1 compression).

After frequency weighting adjustment, the reconstructed image appears to have a better visual quality. The girls in Figure 8.1(a) have more facial skin details, such as freckles, while those details in Figure 8.1(b) are not so clear. Therefore, implementing the frequency weighting adjustment feature to the LZC algorithm has the potential to improve bit-allocation efficiency, which in turn improves the visual quality of reconstructed images.

(a)

(b)

Figure 8.1: Subband frequency weighting adjustment - *Girls* image (a) with subband frequency weighting adjustment (b) without subband frequency weighting adjustment

Y component			
Level	LH	HL	HH
5	1	1	1
4	1	1	1
3	0.999994	0.999994	0.999988
2	0.837755	0.837755	0.701837
1	0.275783	0.275783	0.090078

Cb component			
Level	LH	HL	HH
5	0.812612	0.812612	0.737656
4	0.679829	0.679829	0.567414
3	0.488887	0.488887	0.348719
2	0.267216	0.267216	0.141965
1	0.089950	0.089950	0.737656

Cr component			
Level	LH	HL	HH
5	0.856065	0.856065	0.796593
4	0.749805	0.749805	0.655884
3	0.587213	0.587213	0.457826
2	0.375176	0.375176	0.236030
1	0.166647	0.166647	0.070185

Table 8.1: Frequency weighting for YCbCr color space in JPEG2000

8.2.3 Summary

This section has demonstrated that implementing frequency weighting adjustment can improve the performance of the LZC algorithm. However, there are still three aspects we may need to investigate in the future.

First, the example shown in this section is only for fixed frequency weighting. That is, the frequency weighting adjustment is fixed for a certain viewing condition. However, for different applications the viewing conditions will vary, so the LZC codec will require a function to adjust the frequency weighting accordingly. Also, the frequency weighting set shown in Table 8.1 is only designed for visually lossless or near lossless compression. Therefore, different sets of frequency weights will be required for lossy compression, as well as adjustable sets of frequency weights for visual progressive coding.

Second, although we tested the effect of frequency weighting adjustment using different images, only *Girls* shows a noticeable visual quality improvement. The visual quality improvement for the other images is marginal or even barely recognizable, especially for the natural scenery images which have many high frequency features. This could be due to the set of frequency weights used in the experiment matching the characteristic of *Girls*. Therefore, different sets of frequency weights may be required for images with different characteristics.

Third, the results shown in this section are in YCbCr color space only. This is because the frequency weighting sets are only available in the YCbCr color space. Therefore, for allowing the LZC codec to perform frequency weighting adjustment in the YUV color space, we will need to determine a frequency weighting set by measuring the CSFs of wavelet subbands in the YUV color space.

8.3 Video Layer Separation for 3DLZC

This section suggests a very simple algorithm to separate foreground and background pixels for content based video coding. This algorithm exploits the characteristics of the 3D wavelet transform and will only add a small amount of overhead to 3DLZC.

8.3.1 Video Layer Separation Algorithm

For content based video coding, a binary mask plane called the α-plane is required to indicate the foreground and background regions. Conventionally, an α-plane is constructed by ME/C schemes [188, 189, 190] but this results in higher complexity and is not suitable for the ME/C-free 3DLZC. Alternatively, we can apply simple thresholding and morphological operations to the information available from the existing temporal wavelet transform in the 3DLZC algorithm to construct an α-plane.

As we have mentioned in Section 7.2, if a temporal wavelet transform is applied to a GOF to the highest dyadic level, the high frequency temporal frame (or subband) in the coarsest transform level will contain all of the changes in that GOF, such as the *H3* frame shown in Figure 8.2. We can simply classify the region containing changes as the foreground region and the region containing no changes as the background region. Therefore, based on the information provided by this coarsest high frequency frame, we can construct an α-plane for the 3DLZC codec to locate the foreground and background region in a GOF.

The procedure for constructing an α-plane is shown in Figure 8.3. To construct the α-plane, a

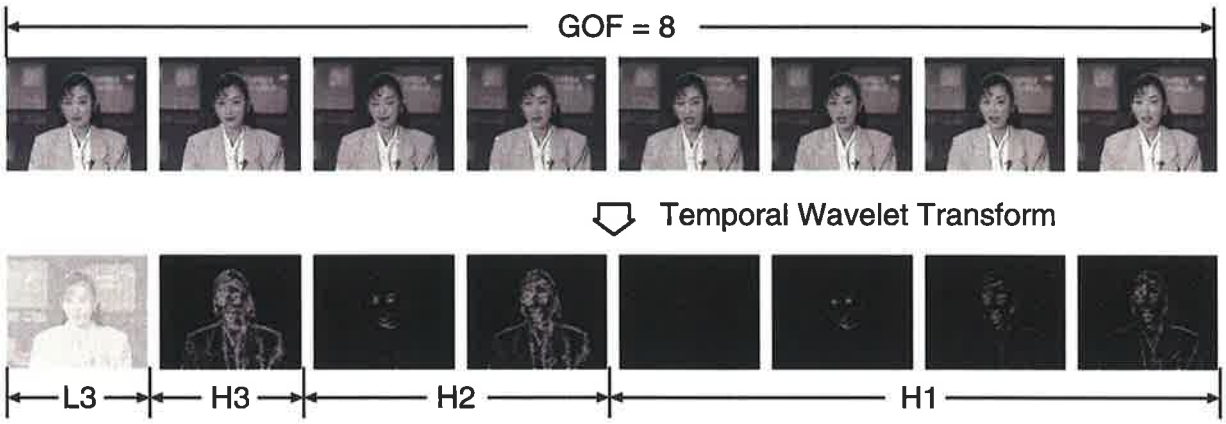


Figure 8.2: Application of temporal wavelet transform on an eight-frame Akiyo GOF

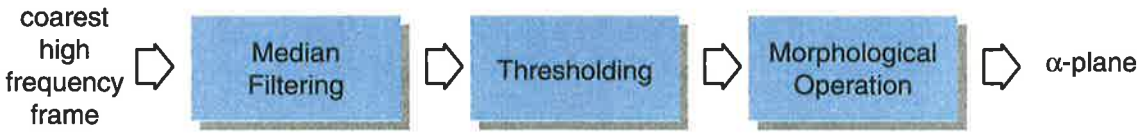


Figure 8.3: α-plane construction procedures

median filter is first applied to the coarsest high frequency frame. Since the function of a median filter is to smooth a group of neighboring pixels according to their median, unwanted noise in this high frequency frame will be eliminated. After that, the α-plane is constructed based on the median filtered frame. That is, if the magnitude of a pixel in this median filtered frame is above a certain threshold, the pixels in the GOF at that spatial location have changed significantly and they are classified as foreground pixels. The pixel in the α-plane at that location is then set to $'1'$. On the other hand, if the magnitude of a pixel is below a certain threshold, then this pixel is assumed to have an insignificant change, and is classified as a background pixel. The pixel in the α-plane at that location is then set to $'0'$.

Figure 8.4(a) shows an example of an α-plane by thresholding the median filtered frame. The foreground region is denoted by white pixels, and the background region is denoted by black pixels. However, the application of thresholding to the filtered frame will not generate an α-plane to reflect the correct shape of the foreground region. For instance, the black regions in Figure 8.4(a) represent the background, but some of the background areas are inside the foreground object. Hence, the morphological *close* operation is applied to the α-plane to obtain a better result.

The morphological close operation [191, 192] is

$$X \bullet S = (X \oplus S) \ominus S \tag{8.1}$$

where set X is the input image and set S is the structuring element with a smaller size than the image and a simple geometrical shape. The symbol $\oplus$ is the dilation operator which performs Minkowski set addition and is defined by

$$X \oplus S(r,c) = max_{(i,j)\in S}\{X(r-i, c-j) + S(i,j)\}. \tag{8.2}$$

Likewise, the symbol $\ominus$ is the erosion operator which performs Minkowski set subtraction and is defined by

$$X \ominus S(r,c) = min_{(i,j)\in S}\{X(r-i, c-j) - S(i,j)\}. \tag{8.3}$$

After applying the morphological close operation to Figure 8.4(a), the black regions inside the foreground object are removed to produce a cleaner α-plane to represent the foreground region and the background region, such as shown in Figure 8.4(b).

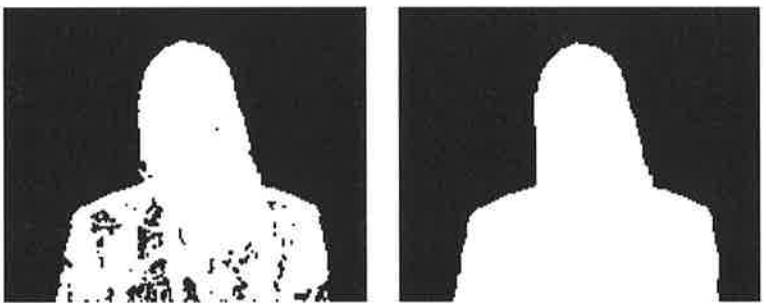


Figure 8.4: (a) α-plane obtained from median filtering; (b) α-plane after applying morphological close operation.

We should note that for each GOF we only need to construct an α-plane for the Y sequence. The α-plane of the Y sequence can then be used as the reference plane to separate the foreground and background regions for the chrominance U and V sequences.

8.3.2 Results

The *Akiyo*, *News*, and *Mother-daughter* video sequences were used to test the video layer separation algorithm. For simplicity, the filter set used for the temporal wavelet transform was the Haar set. The number of frames per GOF for all three test sequences was eight frames. Therefore, the highest temporal wavelet transform level was three. The test video sequences were

re-sampled from 30 frames per second to 10 frames per second by dropping 2 frames from every 3 frames.

The experimental results are shown in Figures 8.5 to 8.7 and can be interpreted as follows. Firstly, although the construction of an α-plane only involves median filtering and the morphological close operation, the results show that this proposed video layer separation algorithm can accurately distinguish the foreground and background regions. In Figure 8.5, the news reporter Akiyo was precisely separated from the background, with a very neat outline. In Figure 8.6, the foreground of the *News* sequence contains the human objects, as well as the background TV screen which constantly changes. Similarly, the outlines of those two reporters were accurately separated from the still background. In Figure 8.7, the human objects have a slightly larger movement than those in the previous two sequences. Therefore, the extracted foreground contains not only the human objects, but also some background pixels around the human objects.

Secondly, since those three test sequences have a fairly still background, the background planes extracted from the first frame of each GOFs can be used as a common background for the other frames in the same GOFs. Figures 8.5(e), 8.6(e), and 8.7(e) illustrate the merge of the common background frame to the last frame in a GOF. Those results show that there is no visual difference between the merged frames and the original frames. Therefore, for achieving better coding results, 3DLZC can maintain and encode one common background frame for each GOF, so that more bit budget can be allocated to the foreground frames.

8.3.3 Summary

In the following we suggest three aspects for improving the proposed video layer separation algorithm in the future.

First, although the size of the GOF in our experiment was eight frames, a sixteen-frame GOF can also be applied to the proposed algorithm. 3DLZC, as well as other 3D wavelet coding algorithms, work better with sixteen-frame GOFs than eight-frame GOFs. Also, better coding results can be achieved by the application of one universal background frame to the entire sixteen frames in the GOF. However, a modification might be necessary to the proposed algorithm to take into account sudden scene changes or more object movements due to the wider frame span. Therefore, instead of using the coarsest high frequency frame as a reference to construct the α-plane, those two high frequency frames in the second coarsest transform level should be used, such as $H2$ in Figure 8.2. The first frame and the second frame in $H2$ represent the changes in the first-half and the second-half of GOF, respectively. Based on these two frames, two sub-α-planes can be constructed first and then be merged to produce the α-plane

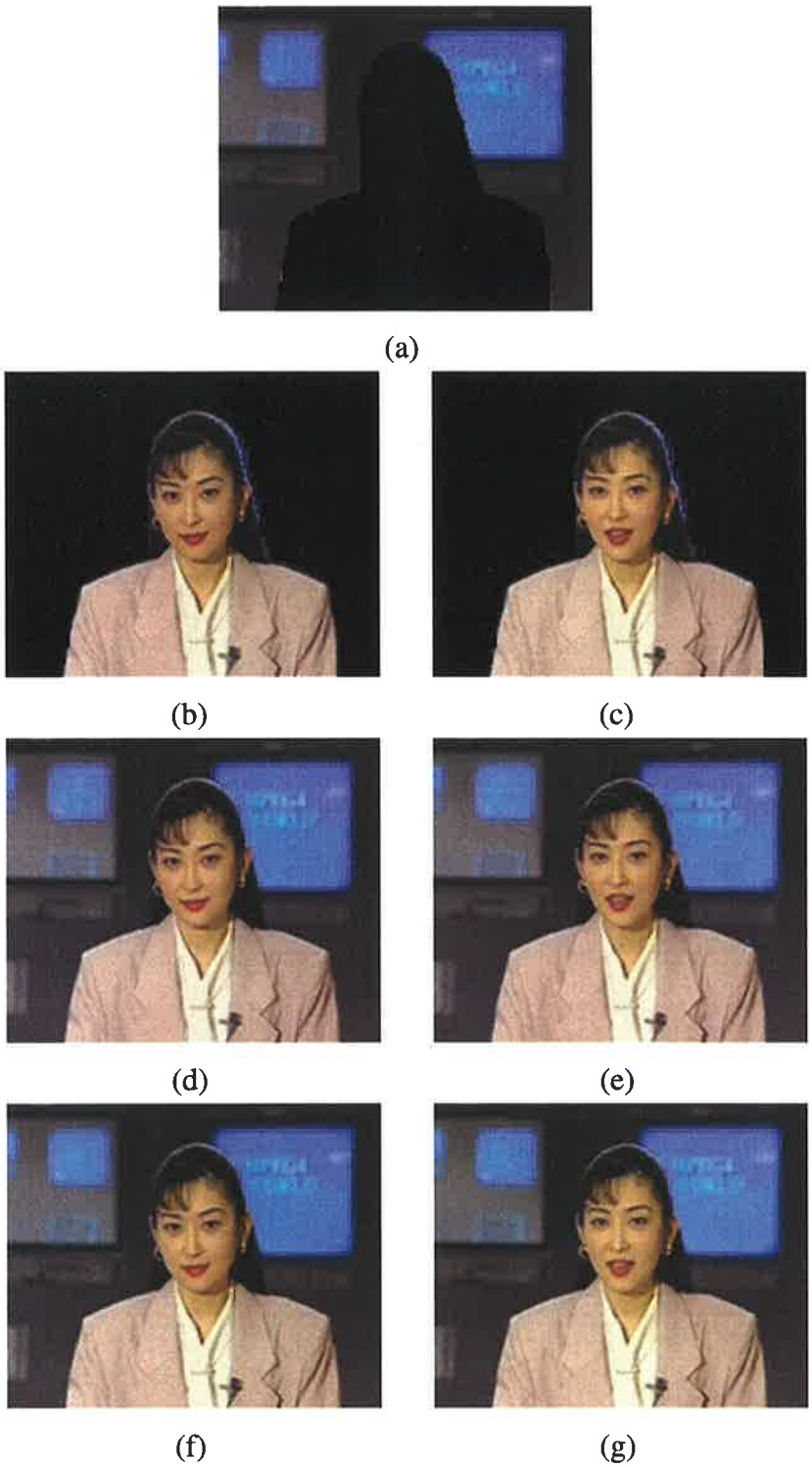


Figure 8.5: Video layer separation - *Akiyo* sequence (a) the common background for a GOF (b) the foreground of frame 000 (c) the foreground of frame 021 (d) merging the foreground of frame 000 to the common background (e) merging the foreground of frame 021 to the common background (f) original frame 000 (g) original frame 021

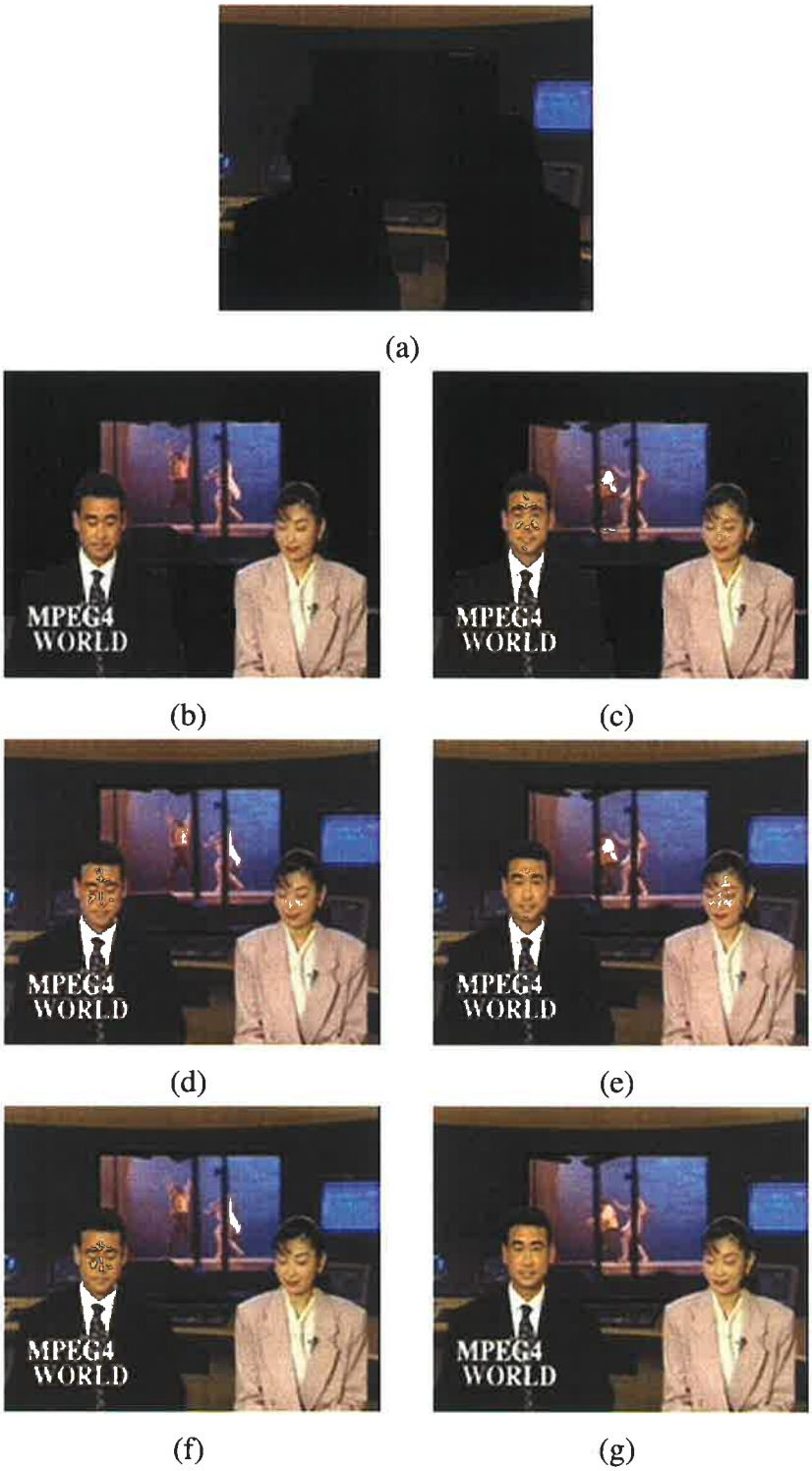


Figure 8.6: Video layer separation - *News* sequence (a) the common background for a GOF (b) the foreground of frame 048 (c) the foreground of frame 069 (d) merging the foreground of frame 048 to the common background (e) merging the foreground of frame 069 to the common background (f) original frame 048 (g) original frame 069

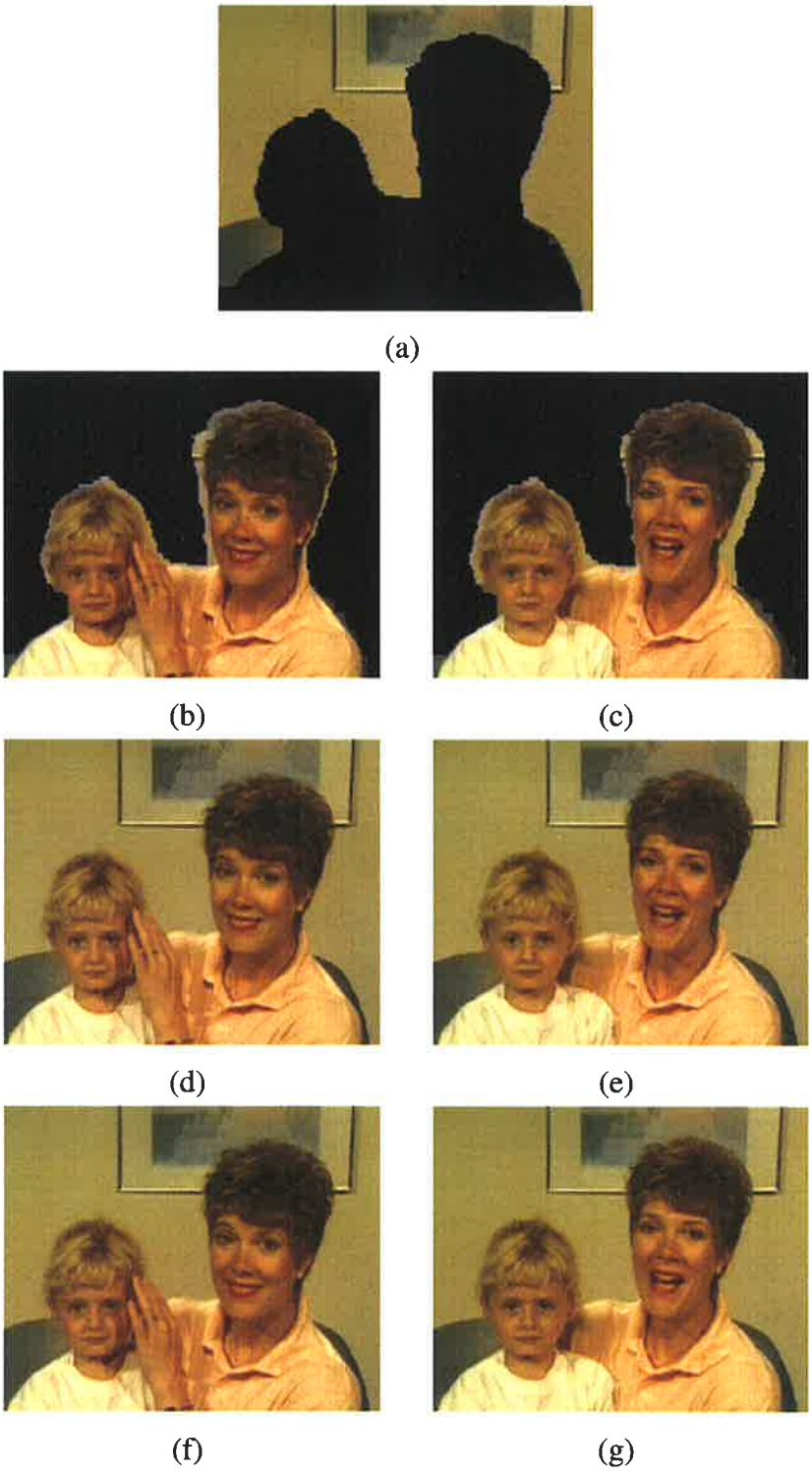


Figure 8.7: Video layer separation - *Mother-daughter* sequence (a) the common background for a GOF (b) the foreground of frame 048 (c) the foreground of frame 069 (d) merging the foreground of frame 048 to the common background (e) merging the foreground of frame 069 to the common background (f) original frame 048 (g) original frame 069

for the whole GOF.

Second, the structuring element S in Equation 8.1 determines the performance of the morphological close operation. If a small geometry S is used, several iterations of the close operation will be required to 'close' the black regions inside the foreground object. However, a smaller S can produce foreground frames with a finer contour. On the other hand, if a large geometry S is used, only one or two applications of the close operation will be enough to close the black regions, but a larger S will causes foreground frames to have a rougher contour.

Third, the video sequences in our experiment were produced under ideal conditions, i.e. no camera noise and no ambient noise. Therefore, the backgrounds of the successive frames are all the same and can easily be detected by the temporal wavelet transform. However, in real-life applications, the background in the video frames will not be as clean as those laboratory test sequences, due to the changes of ambient light intensity and the thermal and quantization noise of the video camera. As a result, the use of the temporal wavelet transform will not be able to resolve a proper α-plane for representing the foreground and background regions. Hence, an additional denoising process, such as the histogram denoising method proposed by Habili et al. [193], will be needed for removing the unwanted noise before the application of the temporal wavelet transform.

In summary, we have suggested a very low complexity algorithm to separate the foreground and background regions in a GOF. This algorithm exploits the information that has already been calculated by the temporal wavelet transform and only involves a filtering, a thresholding and several morphological operations. Because of its simplicity, this algorithm will allow 3DLZC not only to perform content based coding but also to possibly perform it in real-time.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

In this thesis, we began our discussion in Chapter 1 with two simple questions about rate-distortion optimization for image and video compression. The first question is *"What types of information are more important than others for minimizing distortion?"* and the second question is *"How can we effectively encode those important types of information using the least amount of coding bit-budget?"* We have tried to answer these two questions in the rest of this thesis. We also have focused our discussion on image and video compression on the zerotree scheme only. In the following we will summarize the contributions of this thesis.

We have two ways to answer the first question. A straightforward way to seek the answer is from the mathematical point of view, as it is more closely related to engineering and would make engineers feel more comfortable. However, in this thesis we have also tried to look for the answer from the psychophysical and psychological point of view. Therefore, in Chapter 2 we have investigated some human visual system characteristics that are related to image compression. In Chapter 3 we have discussed how those characteristics can be exploited in some of the key elements for image compression.

We then tried to answer the second question. Therefore, in Chapter 4 we investigated some commonly used visual data compression schemes. Because quality assessment and image compressibility are also important in determining compression performance, we have discussed them in the same chapter.

Among the compression schemes investigated, the multiresolution methods which exploit zerotree theory not only provide a good answer to the second question, but also have a number of desirable features, such as low coding complexity, quality embededness, and precise

rate-distortion control. However, as we have identified in Chapter 5, the current zerotree compression algorithms suffer from high coding memory requirements that make them not suitable for hardware implementation.

Therefore, in this thesis we have proposed the LZC algorithm to solve the problem of high coding memory requirement. Since the proposed LZC algorithm is also based on zerotree theory, it not only lowers the coding memory significantly but also inherits all of the benefits of zerotree coding schemes. Besides the significant reduction of coding memory requirement, the LZC algorithm has also further reduced the coding complexity by using just one coding pass.

We have extensively discussed the proposed LZC algorithm for color image compression in Chapter 6. In this chapter we have also shown how we can make use of the characteristics of the human visual system in the the LZC algorithm for improving the bit-allocation efficiency, and in turn improving coding results. From the experimental results we have shown that the LZC algorithm can perform as well as the benchmark zerotree coding algorithm SPIHT and the new compression standard JPEG2000.

We have extended the LZC algorithm for color video coding in Chapter 7. The video coding version of the LZC algorithm is called the 3DLZC algorithm. In order to minimize the coding complexity, we exploit the 3D wavelet transform for temporal data decorrelation in 3DLZC and use the 3DLZC data structure to encode the video data. The experimental results have shown that the performance of the 3DLZC algorithm is comparable to the low bit-rate video compression standard, H.263+ and 3D-SPIHT.

At the end of this thesis we have suggested two possible extensions for the LZC algorithm in Chapter 8. One is the implementation of frequency weighting in LZC, and another is the implementation of video layer separation feature in 3DLZC. We have been able to demonstrate that by implementing the frequency weighting feature in LZC we can improve the compression image quality in at least some cases. However, we have only been able to propose video content separation algorithm but not to implement it in 3DLZC.

In short, the problems associated with image and video compression are still within the scope of the classic rate-distortion optimization and still must be solved by answering those two questions we have asked ourselves in this thesis. Many image and video compression algorithms would have been proposed or developed by researchers during the time of writing this thesis. However, in this thesis we are trying to answer those two questions from a different direction. We are not trying to provide optimal solutions to those two questions. Rather, we are trying to provide alternative answers to them. LZC and 3DLZC algorithm will still require more refinement to make them perform better. Thus, we should conclude this thesis by suggesting some future work on the LZC and 3DLZC algorithms.

9.2 Future Work

We have demonstrated the viability of the LZC and TPLZC algorithms and shown that they have performance that is better than, or comparable to, other published algorithms, with lower complexity. There are other aspects of this algorithm that could be investigated further, but beyond the scope of this thesis. These avenues for further investigation are outlined below.

9.2.1 For the LZC Algorithm

Image size scaling: The LZC algorithm presented in this thesis only provides the functionality for compression quality scaling. However, it would also be possible to implement the functionality of image size scaling. That is, at low coding bit-rate the image can either be reconstructed to a poor quality or to a smaller size, and after the bit-rate increases, the image can then be reconstructed to a better quality and to a bigger size.

Wavelet packets: In this thesis, we only use the Mallat wavelet decomposition to decorrelate the image spatial redundancy. However, in future the wavelet packet could also be exploited for image data decorrelation. At the cost of higher implementation complexity, the wavelet packet has a better frequency decomposition ability, so the image data can be more effectively decorrelated, giving better compression results [194].

Region of interest coding: The function that allows users to select the region of interest in an image could also be possibly implemented in the LZC algorithm. That is, users can specify one or more regions in an image to be coded at a higher quality, and then the LZC codec can allocate more bit-budget to those regions to improve the reconstructed visual quality.

Watermarking: In order to protect the ownership of the images, a watermarking function could be possibly implemented in the LZC algorithm. Similarly to image compression, watermarking could be implemented in the frequency domain, as well as based on the characteristics of our visual system so that the watermark can be embeded in the image without visual quality degradation. Therefore, these similarities could be exploited to implement watermarking functionality to LZC algorithm.

Hardware Implementation: Since one of the motivations for deriving the LZC algorithm is for hardware implementation, the LZC algorithm could be implemented in hardware to verify our presumption that LZC is a hardware implementation friendly image compression algorithm.

9.2.2 For the 3DLZC Algorithm

Raster tree search: This thesis only presents the recursive tree search 3DLZC algorithm. Therefore, the 3DLZC algorithm could be modified to perform raster tree search, like the TPLZC algorithm.

Content base coding: In this thesis we only present a simple video layer separation algorithm without implementing it in the 3DLZC algorithm. Therefore, in the future this layer separation algorithm could be implemented in the 3DLZC for content base video coding.

Video frame size scaling: Similarly to LZC, the 3DLZC algorithm presented in this thesis only allows visual quality scaling, but not video frame size scaling. Therefore, the functionality of video frame size scaling could be implemented in the 3DLZC algorithm for heterogeneous distribution channels, such as the Internet.

Motion estimation and compensation: In the 3DLZC algorithm, the reduction of temporal data redundancy is only performed by the temporal wavelet transform. However, the temporal wavelet transform is only suitable for decorrelating low motion video sequences. Therefore, for coding high motion video sequences, motion estimation and compensation could also be implemented in the 3DLZC algorithm.

Hardware implementation: Similarly, in the future the 3DLZC algorithm could also be implemented in hardware to verify its performance.

Appendix A

Test Images

The test images used for the performance evalution of the LZC algorithm are presented here. The test image information is summarized in Table A.1, and the test images are presented in Figure A.1 to Figure A.7.

Test Images	Color Depth (bits)	Dimension (rowxcolumn)	Max. Dyadic Level
Mandrill	24	512x512	9
Girls	24	512x512	9
Lena	24	512x512	9
Peppers	24	512x512	9
Bike	24	512x512	9
Cacti	24	448x704	6
Jellybeans	24	256x384	7
Matsuri	24	480x640	5
Railway	24	480x640	5
Flowers	24	480x640	5
Flowerfield1	24	480x640	5
Flowerfield2	24	480x640	5
Mt Fuji1	24	480x640	5
Mt Fuji2	24	480x640	5
Mt Fuji3	24	480x640	5
Farm	24	480x640	5
Jetty	24	480x640	5

Table A.1: The color depth, dimensions and maximum available wavelet dyadic transform levels of test images

(a) *Mandrill*

(b) *Girls*

(c) *Lena*

(d) *Peppers*

Figure A.1: Original test images

(a) *Bike*

(b) *Jellybeans*

(c) *Cacti*

Figure A.2: Original test images - continued

(a) *Matsuri*

(b) *Railway*

Figure A.3: Original test images - continued

(a) *Flowers*

(b) *Flowerfield1*

Figure A.4: Original test images - continued

(a) *Flowerfield2*

(b) *Mt Fuji1*

Figure A.5: Original test images - continued

(a) *Mt Fuji2*

(b) *Mt Fuji3*

Figure A.6: Original test images - continued

(a) *Farm*

(b) *Jetty*

Figure A.7: Original test images - continued

Bibliography

[1] C.E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423,623–656, 1948. http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html.

[2] C.E. Shannon. Coding Theorems For a Discrete Source with a Fidelity Criterion. *IRE Nat. Conv. Rec.*, 4:142–163, 1959.

[3] T. Berger. *Rate Distortion Theory: A Mathematical Basis for Data Compression.* Prentice-Hall, 1971.

[4] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen. Low-Power CMOS Digital Design. *IEEE Journal of Solid-State Circuit*, 27(4):473–483, April 1992.

[5] A.P. Chandrakasan and R.W. Brodersen. Minimizing Power Consumption in Digital CMOS Circuits. *Proceeding of the IEEE*, 83(4):498–523, April 1995.

[6] L.S. Nielsen. *Low-Power Asynchronous VLSI Design.* PhD thesis, Technical University of Denmark, Lyngby, 1997.

[7] L. Chandrasena and M.J. Liebelt. A Rate Selection Algorithm for Quantized Undithered Dynamic Supply Voltage Scaling. In *Proceedings of the 2000 International Symposium on Low Power Electronics and Design*, pages 213–215, Italy.

[8] D.H. Hubel. *Eye, Brain, and Vision.* Scientific American Library, 1988.

[9] B.A. Wandell. *Foundations of Vision.* Sinauer Associates, Massachusetts, 1995.

[10] M. Livingstone and D. Hubel. Segregation of Form, Color, Movement, and Depth: Anatomy, Physiology, and Perception. *Science*, 240(4853):740–749, 1988.

[11] R.C. Gonzalez and R.E. Woods. *Digital Image Processing.* Addison-Wesley Publishing Company, 1992.

[12] A.K. Jain. *Fundamentals of Digital Image Processing.* Prentice-Hall, 1989.

[13] S. Winkler. Issues in Vision Modeling for Perceptual Video Quality Assessment. *Signal Processing*, 78(2):231–252, October 1999.

[14] J. Lubin. *Digital Images and Human Vision*, chapter 13 The Use of Psychophysical Data and Models in the Analysis of Display System Performance, pages 161–178. MIT Press, 1993.

[15] S. Appelle. Perception and Discrimination as a Function of Stimulus Orientation: The Oblique Effect in Man and Animals. *Psychological Bulletin*, 78(4):266–278, 1972.

[16] R.J. Mansfield. Neural Basis of Orientation Perception in Primate Vision. *Science*, 186:1133–1134, 1974.

[17] R.L. De Valois, E.W. Yund, and N. Hepler. The Orientation and Direction Selectivity of Cells in Macaque Visual Cortex. *Vision Research*, 22(5):531–544, 1982.

[18] D.M. Coppola, L.E. White, D. Fitzpatrick, and D. Purves. Unequal Representation of Cardinal and Oblique Contours in Ferret Visual Cortex. *Proceedings of the National Academy of Science of the United States of America*, 95:2621–2623, March 1988.

[19] C.S. Furmanski and S.A. Engel. An Oblique Effect in Human Primary Visual Cortex. *Natural Neuroscience*, 3(6):535–536, June 2000.

[20] C. Zetzsche, E. Barth, and B. Wegmann. *Digital Images and Human Vision*, chapter 10 The Importance of Intrinsically Two-dimensional Image Features in Biological Vision and Picture Coding, pages 109–140. MIT Press, 1993.

[21] M.P. Eckert. Perceptual Quality Metrics Applied to Still Image Compression. *Signal Processing*, 70(3):177–200, November 1998.

[22] M.J. Nadenau, J. Reichel, and M. Kunt. Performance Comparison of Masking Model Based on a New Psychovisual Test Method with Natural Scenery Stimuli. *Submitted to IEEE Transactions on Image Processing* , November 2000. http://dewww.epfl.ch/ nadenau/Research/publication.htm.

[23] G.E. Legge and J.M. Foley. Contrast Masking in Human Vision. *Journal of Optical Society of America*, 70(12):1458–1471, 1980.

[24] J.A. Solomon, A.B. Watson, and A. Ahumada. Visibility of DCT Basis Functions: Effects on Contrast Masking. In *Data Compression Conference*, pages 361–370, 1994. http://vision.arc.nasa.gov/publications.

[25] J.M. Foley. Human Luminance Pattern Mechanisms: Masking Experiments Require a New Model. *Journal of Optical Society of America A*, 11(6):1710–1719, June 1994.

[26] P.C. Teo and D.J. Heeger. Perceptual Image Distortion. In *Proceedings of SPIE*, volume 2179, pages 127–141, 1994. http://white.stanford.edu/ heeger/publications.html.

[27] A.B. Watson, J.A. Solomom, and A.J. Ahumada Jr. Visibility of DCT Basis Functions: Effects of Display Resolution. In *Data Compression Conference*, pages 371–379, 1994. http://vision.arc.nasa.gov/publications.

[28] G.E. Legge, D. Kersten, and A.E. Burgess. Contrast Discrimination in Noise. *Journal of Optical Society of America*, 4(2):391–404, February 1987.

[29] H. Mostafavi and D.J. Sakrison. Structure and Properties of a Single Channel in the Human Visual System. *Vision Research*, 16:957–968, 1976.

[30] D.J. Swift and R.A. Smith. Spatial Frequency Masking and Weber's Law. *Vision Research*, 23:495–506, 1983.

[31] K.T. Blackwell. The Effect of White and Filtered Noise on Contrast Detection Threshold. *Vision Research*, 38(2):267–280, 1998.

[32] A.E. Burgess, X. Li, and C.K. Abbey. Visual Signal Detectability with Two Noise Components: Anomalous Masking Effects. *Journal of Optical Society of America A*, 14(9):2420–2442, 1997.

[33] M.P. Eckstein, Jr. A.J. Ahumada, and A.B. Watson. Visual Signal Detection in Structured Backgrounds. II. Effects of Contrast Gain Control, Background Variations, and White Noise. *Journal of Optical Society of America A*, 14(9):2406–2419, 1997. http://vision.arc.nasa.gov/publications.

[34] A.B. Watson, R. Borthwick, and M. Taylor. Image Quality and Entropy Masking. volume 3016, pages 358–371, 1997. http://vision.arc.nasa.gov/publications.

[35] W.E. Glenn. *Digital Images and Human Vision*, chapter 6 Digital Image Compression Based on Visual Perception, pages 63–71. MIT Press, 1993.

[36] T.N. Cornsweet. *Visual Perception*. Academic Press, New York, 1970.

[37] D.H. Kelly. Motion and Vision. I. Stabilized Images of Stationary Gratings. *Journal of Optical Society of America*, 69(9):1266–1274, September 1979.

[38] D.H. Kelly. Motion and Vision. II. Stabilized Spatio-Temporal Threshold Surface. *Journal of Optical Society of America*, 69(10):1340–1349, October 1979.

[39] D.H. Kelly. Spatiotemporal Variation of Chromatic and Achromatic Contrast Threshold. *Journal of Optical Society of America*, 73(6):742–750, June 1983.

[40] J. Yang and W. Makous. Spatiotemporal Separability in Contrast Sensitivity. *Vision Research*, 34(19):2569–2576, 1994.

[41] R.F. Hess and R.J. Snowden. Temporal Properties of Human Visual Filters: Number, Shapes and Spatial Covariation. *Vision Research*, 32(1):47–59, 1992.

[42] R.E. Fredericksen and R.F. Hess. Estimating Multiple Temporal Mechanisms in Human Vision. *Vision Research*, 38(7):1023–1040, 1998.

[43] A.N. Netravali and B.G. Haskell. *Digital Pictures: Representation, Compression, and Standards*. Plenum Press, New York, 195.

[44] A.J. Seyler and Z.L. Budrikis. Detail Perception after Scene Changes in Television Image Presentations. *IEEE Transactions on Information Theory*, 11(1):31–43, January 1965.

[45] Jr. A.J. Ahumada, B.L. Beard, and R. Eriksson. Spatio-Temporal Discrimination Model Predicts Temporal Masking Functions. In *Proceedings of SPIE: IS&T/SPIE Electronic Imaging Symposium*, pages 24–30, San Jose, January 1998. http://vision.arc.nasa.gov/publications.

[46] O.H. Schade. Optical and Photoelectric Analog of the Eye. *Journal of Optical Society of America*, 46(9):721–739, 1956.

[47] J.L. Mannos and D.J. Sakrison. The Effects of a Visual Fidelity Criterion on the Encoding of Images. *IEEE Transactions on Information Theory*, 20(4):525–536, 1974.

[48] F.X.J. Lukas and Z.L. Budrikis. Picture Quality Prediction Based on a Visual Model. *IEEE Transactions on Communications*, 30(7):1679–1692, July 1982.

[49] D.H. Hubel and T.N. Wiesel. Receptive Fields and Functional Architecture of Monkey Striate Cortex. *Journal of Physiology*, 195:215–243, 1968.

[50] J.G. Daugman. Two-Dimensional Spectral Analysis of Cortical Receptive Field Profiles. *Vision Research*, 20(10):847–856, 1980.

[51] A.B. Watson and Jr. A.J. Ahumada. A Hexagonal Orthogonal-Oriented Pyramid as a Model of Image Representation in Visual Cortex. 36(1):97–106, January 1989.

[52] J.P. Jones and L.A. Palmer. The Two-Dimensional Spatial Structure of Simple Receptive Fields in Cat Striate Cortex. *Journal of Neurophysiology*, 58(6):1187–1211, December 1987.

[53] J.A. Movshon, I.D. Thompson, and D.J. Tolhurst. Spatial Summation in the Receptive Fields of Simple Cells in the Cat's Striate Cortex. *Journal of Physiology*, 283:53–77, 1978.

[54] R.L. De Valois, D.G. Albrecht, and L.G. Thorell. Spatial Frequency Selectivity of Cells in Macaque Visual Cortex. *Vision Research*, 22(5):545–559, 1982.

[55] J.P. Jones, A. Stepnoski, and L.A. Palmer. The Two-Dimensional Spectral Structure of Simple Receptive Fields in Cat Striate Cortex. *Journal of Neurophysiology*, 58(6):1212–1232, December 1987.

[56] G.C. Philips and H.R. Wilson. Orientation Bandwidth of Spatial Mechanisms Measured by Masking. *Journal of Optical Society of America A*, 1(2):226–232, 1984.

[57] J.H. van Hateren and D.L. Ruderman. Independent Component Analysis of Natural Image Sequences Yields Spatio-Temporal Filters Similar to Simple Cells in Primary Visual Cortex. *Proceedings of Royal Society of London*, B-265:1–8, 1988.

[58] H.R. Wilson and D. Regan. Spatial Frequency Adaption and Grating Discrimination: Predictions of a Line-Element Model. *Journal of Optical Society of America A*, 1(2):1091–1096, 1984.

[59] A.B. Watson. The Cortex Transform: Rapid Computation of Simulated Neural Images. *Computer Vision, Graphics, and Image Processing*, 39:311–327, 1987. http://vision.arc.nasa.gov/publications.

[60] S. Daly. *Digital Images and Human Vision*, chapter 14 The Visible Difference Predictor: An Algorithm for the Assessment of Image Fidelity, pages 179–206. MIT Press, 1993.

[61] R.M. Evans. *The Perception of Color*. John Wiley & Sons, New York, 1974.

[62] G.A. Agoston. *Color Theory and Its Application in Art and Design*. Springer-Verlag, Berlin, 1979.

[63] S.L. Merbs and J. Nathans. Absorption Spectra of Human Cone Pigments. *Nature*, 356:433–435, April 1992.

[64] A. Stockman, L.T. Sharpe, and C. Fach. The Spectral Sensitivities of the Human Short-wavelength Sensitive Cones Derived from Threshold and Color Matches. *Vision Research*, 39:2901–2927, 1999.

[65] A. Stockman and L.T. Sharpe. The Spectral Sensitivities of the Middle- and Long-wavelength Cones Derived from Measurements in Observers of Known Genotype. *Vision Research*, 40:1711–1737, 2000.

[66] D.B. Judd and G. Wyszecki. *Color in Business, Science and Industry*. John Wiley & Sons, New York, 1975.

[67] D.M. Dacey. Circuitry for Color Coding in the Primate Retina. *Proceedings of the National Academy of Science of the United States of America*, 93:582–588, January 1996.

[68] D. Jameson and L.M. Hurvich. Some Quantitative Aspects of an Opponent-Colors Theory. I. Chromatic Response and Spectral Saturation. *Journal of Optical Society of America*, 45(7):546–552, July 1955.

[69] L.M. Hurvich and D. Jameson. Some Quantitative Aspects of an Opponent-Colors Theory. II. Brightness, Saturation, and Hue in Normal and Dischromatic Vision. *Journal of Optical Society of America*, 45(8):602–616, August 1955.

[70] G. Buchsbaum. *Digital Images and Human Vision*, chapter 9 Visual System Considerations in the Coding of Natural Color Images, pages 99–108. MIT Press, 1993.

[71] G. Buchsbaum and A. Gottschalk. Trichromacy, Opponent Colours Coding and Optimum Colour Information Transmission in the Retina. *Proceedings of Royal Society of London*, B-220:89–113, 1983.

[72] D.L. Ruderman, T.W. Cronin, and C.-C. Chiao. Statistics of Cone Responses to Natural Images: Implications for Visual Coding. *Journal of Optical Society of America A*, 15(8):2036–2045, August 1998.

[73] G. Wyszecki and W.S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulae*. John Wiley & Sons, 1982.

[74] A.M. Derrington, J. Krauskopf, and P. Lennie. Chromatic Mechanisms in Lateral Geniculate Nucleus of Macaque. *Journal of Physiology*, 357:241–265, 1984.

[75] K.T. Mullen. The Contrast Sensitivity of Human Colour Vision to Red-Green and Blue-Yellow Chromatic Gratings. *Journal of Physiology*, 359:381–400, 1985.

[76] M.J. Nadenau and J. Reichel. Opponent Color, Human Vision and Wavelets for Image Compression. In *Proceedings of the 7th Color Imaging Conference*, pages 237–242, Scottsdale, Arizona, November 1999. http://dewww.epfl.ch/ nadenau/Research/publication.htm.

[77] S.L. Guth. Model for Color Vision and Light Adaptation. *Journal of Optical Society of America A*, 8(6):976–993, June 1991.

[78] J.J. Atick, Z. Li, and A.N. Redlich. Understanding Retinal Color Coding from First Principles. *Neural Computation*, 4:559–572, 1992.

[79] R.L. De Valois and K.K. De Valois. A Multi-Stage Color Model. *Vision Research*, 33(8):1053–1065, 1993.

[80] CIE Technical Committee TC-1.3. *Colorimetry*. Publication CIE, Vienna, 1986.

[81] Recommendation ITU-R BT.709-4. *Parameter Values for the HDTV Standards for Production International Programme Exchange*. ITU, Geneva, 1995.

[82] Recommendation ITU-R BT.601-5. *Studio Encoding Parameters of Digital Television for Standard 4:3 and Wide-Screen 16:9 Aspect Ratios*. ITU, Geneva, 1995.

[83] L. Gaudart, J. Crebassa, and J.P. Petrakian. Wavelet Transform in Human Visual Channels. *Applied Optics*, 32(22):41194127, August 1993.

[84] S. Mallat. Wavelet for a Vision. *Proceeding of the IEEE*, 84(4):604–614, April 1996.

[85] M. Unser and A. Aldroubi. A Review of Wavelets in Biomedical Applications. *Proceeding of the IEEE*, 84(4):626–638, April 1996.

[86] O. Rioul and M. Vetterli. Wavelets and Signal Processing. *IEEE Signal Processing Magazine*, 8(4):14–38, October 1991.

[87] A. Cohen and J. Kovačević. Wavelets: The Mathematical Background. *Proceedings of the IEEE*, 84(4):514–522, April 1996.

[88] I. Daubechies. Orthonormal Bases of Compactly Supported Wavelets. *Communications on Pure and Applied Mathematics*, 41:909–996, July 1988.

[89] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image Coding Using Wavelet Transform. *IEEE Transactions on Image Processing*, 1(2):205–220, April 1992.

[90] M. Vetterli and J. Kovačević. *Wavelets and Subband Coding*. Prentice Hall PTR, New Jersey, 1995.

[91] G. Strang and T. Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Massachusetts, 1996.

[92] K.R. Castleman. *Digital Image Processing*, chapter Chapter 14: Wavelet Transforms. Prentice Hall, Englewood Cliffs, N.J., 1996.

[93] J.W. Woods and S.D. O'Neil. Subband Coding of Image. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(5):1278–1288, October 1986.

[94] M.J.T. Smith and S.L. Eddins. Analysis/Synthesis Techniques for Subband Image Coding. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(8):1446–1456, August 1990.

[95] S.A. Matucci and R.M. Mersereau. The Symmetric Convolution Approach to the Nonexpansive Implementation. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages V65–V68, Minneapolis, USA, April 1993.

[96] S.A. Matucci. Symmetric Convolution and the Discrete Sine and Cosine Transform. *IEEE Transactions on Signal Processing*, 42(5):1038–1051, 1994.

[97] H. Kiya, K. Nishikawa, and M. Iwahashi. A Development of ymmetric Extension Method for Subband Image Coding. *IEEE Transactions on Image Processing*, 3(1):78–81, January 1994.

[98] R.H. Bamberger, S.L. Eddins, and V. Nuri. Generalized Symmetric Extension for Size-Limited Multirate Filter Banks. *IEEE Transactions on Image Processing*, 3(1):82–87, January 1994.

[99] C. M. Brislawn. Preservation of Subband Symmetry in Multirate Signal Coding. *IEEE Transactions on Signal Processing*, 43(12):3046–3050, December 1995.

[100] C. M. Brislawn. Classification of Nonexpansive Symmetric Extension Transforms for Multirate Filter Banks. *Applied and Computational Harmonic Analysis*, 3:337–357, 1996.

[101] J.N. Bradley and C.M. Brislawn. The FBI Wavelet/Scalar Quantization Standard for Gray-Scale Fingerprint Image Compression. In *Proceedings of SPIE*, volume 1961, pages 293–304, 1993.

[102] O. Rioul. On the Choice of "Wavelet" Filters for Still Image Compression. In *Proceeding of IEEE Data Compression Conference*, pages 550–553, March 1993.

[103] J.D. Villasenor, B. Belzer, and J. Liao. Wavelet Filter Evalution for Image Compression. *IEEE Transactions on Image Processing*, 4(8):1053–1060, August 1995.

[104] D.A. Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, 40(9):1098–1101, September 1952.

[105] J.J Rissanen. Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, 20(5):198–203, May 1976.

[106] R. Pasco. *Source Coding Algorithm for Fast Data Compression.* PhD thesis, Department of Electrical Engineering, Stanford University, Stanford, California, 1976.

[107] I.H. Witten, R.M. Neal, and J.G. Cleary. Arithmetic Coding for Data Compression. *Communications of the ACM*, 30(6):520–540, June 1987.

[108] D. Hankerson, G.A. Harris, and P.D. Johnson Jr. *Introduction to Information Theory and Data Compression*, chapter 6. Arithmetic Coding. CRC Press, 1998.

[109] A. Gersho. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, 1992.

[110] N.S. Jayant and P. Noll. *Digital Coding of Waveforms.* Prentice-Hall, New Jersey, 1984.

[111] K.R. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages and Applications.* Academic Press, San Diego, 1990.

[112] G.K. Wallace. The JPEG Still Picture Compression Standard. *Communications of ACM*, 34(4):30–44, April 1991.

[113] W.B. Pennebaker and J.L. Mitchell. *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold, New York, 1993.

[114] W. Kou. *Digital Image Compression: Algorithms and standards.* Kluwer Academic Publishers, Boston, 1995.

[115] V. Bhaskaran and K. Konstantinides. *Image and Video Conpression Standards: Algorithms and Architectures.* Kluwer Academic Publishers, Massachusettes, 1995.

[116] J.L. Mitchell, W.B Pennebaker, and C.E. Fogg. *MPEG video : compression standard.* Chapman & Hall, New York, 1996.

[117] M.F. Barnsley and A.E. Jacquin. Applications of Recurrent Iterated Function Systems to Images. In *Proceeding of SPIE*, volume 1001, pages 122–131, 1988.

[118] M.F. Barnsley. *Fractal Image Compression.* AK Peters, Ltd., Massachusetts, 1993.

[119] A.E. Jacquin. A Novel Fractal Block-Coding Technique for Digital Images. In *Proceedings of ICASSP'90*, pages 18–30, 1990.

[120] A.E. Jacquin. Image Coding Based on A Fractal Theory of Iterated Contractive Image Transformation. *IEEE Transactions on Image Processing*, 1(1):18–30, January 1992.

[121] M.F. Barnsley and S. Demko. Iterated Function Systems and the Global Construction of Fractals. *Proceedings of the Royal Society London*, A339:243–275, 1985.

[122] I.K. Levy. *Self-Similarity and Wavelet Transforms for the Compression of Still Image and Video Data*. PhD thesis, The University of Warwick, 1998.

[123] I.K. Levy and R. Wilson. Three-Dimensional Wavelet Transform Video Coding Using Symmetric Codebook Vector Quantization. *IEEE Transactions on Image Processing*, 10(3):470–475, March 2001.

[124] H. Gharavi and A. Tabatabai. Sub-Band Coding of Monochrome and Color Images. *IEEE Transactions on Circuits and Systems*, 35(2):207–214, February 1988.

[125] S.G. Mallat. A Theory for Multiresolution Signal Decomposition: the Wavelet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.

[126] A.S. Lewis and G. Knowles. Image Compression Using the 2-D Wavelet Transform. *IEEE Transactions on Image Processing*, 1(2):244–250, April 1992.

[127] J.M. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, December 1993.

[128] J.M. Shapiro. Application of the Embedded Wavelet Hierarchical Image Coder to Very Low Bit Rate Image Coding. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages V–559–V–561, April 1993.

[129] G.L.Fredendall and W.L. Behrend. Picture Quality-Procedures for Evaluating Subjective Effects of Interference. *Proceeding of the IRE*, pages 1030–1034, June 1960.

[130] Recommendation ITU-R BT.500-10. *Methodology for the Subjective Assessment of the Quality of Television Pictures*. ITU, Geneva, 2000.

[131] B. Girod. *Digital Images and Human Vision*, chapter 15 What's Wrong with Mean-squared Error?, pages 207–220. MIT Press, 1993.

[132] Jr. A.J. Ahumada. Computational Image Quality Metrics: A Review. *SID Symposium Digest*, 24:305–308, 1993.

[133] Jr. A.J. Ahumada. Simplified Vision Models for Image Quality Assessment. *SID Symposium Digest*, 27:397–400, 1996.

[134] C. Lambrecht and J.E. Farrell. Perceptual Quality Metric for Digitally Coded Color Images. In *Proceedings of European Signal Processing Conference*, pages 1175–1178, Trieste, Italy, September 1996.

[135] T.N. Pappas and R.J. Safranek. Perceptual Criteria for Image Quality Evaluation, March 1999. http://citeseer.nj.nec.com/14738.html.

[136] D.J. Sakrison. On the Role of the Observer and a Distortion Measure in Image Transmission. *IEEE Transactions on Communications*, COM-25(11):1251–1267, November 1977.

[137] O.D. Faugeras. Digital Color Image Processing within the Framework of a Human Visual Model. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(4):380–393, 1979.

[138] J. Lubin and D. Fibush. Sarnoff jnd vision model. IEEE G-2.1.6 Compression and Processing Subcommittee, August 1997.

[139] N. Jayant. Signal Compression: Technology Targets and Research Directions. *IEEE Journal on Selected Areas in Communications*, 10(5):796–818, June 1992.

[140] N. Jayant, J. Johnston, and R. Safranek. Signal Compression Based on Models of Human Perception. *Proceedings of the IEEE*, 84(10):1385–1422, October 1993.

[141] A.B. Watson. DCT Quantization Matrices Visually Optimized for Individual Images. In *Proceedings of SPIE: Human Vision, Visual Processing, and Digital Display IV*, volume 1913, pages 202–216, 1993. http://vision.arc.nasa.gov/publications.

[142] A.B. Watson, J.A. Solomom, and A.J. Ahumada Jr. Visibility of DCT Basis Functions: Effects of Display Resolution. In *Data Compression Conference*, pages 371–379, 1994. http://vision.arc.nasa.gov/publications.

[143] A.P. Bradley. A Wavelet Visible Difference Predictor. *IEEE Transactions on Image Processing*, 8(5):717–730, May 1999.

[144] A.B. Watson, G.Y. Yang, J.A. Solomon, and J. Villasenor. Visibility of Wavelet Quantization Noise. *IEEE Transactions on Image Processing*, 6(8):1164–1175, August 1997. http://vision.arc.nasa.gov/publications.

[145] W.R. Zettler, J. Huffman, and D.C.P. Liden. Application of Compactly supported Wavelets to Image Compression. In *Proceedings of SPIE*, volume 1244, pages 150–160, 1990.

[146] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image Coding Using Vector Quantization in the Wavelet Transform Domain. volume 4, pages 2297–2300, April 1990.

[147] S. Mallat and S.Zhong. Compact Image Coding from Edges with Wavelet. volume 4, pages 2745–2747, May 1991.

[148] R.A. Devore, B. Jawerth, and B.J. Lucier. Image Compression Through Wavelet Transform Coding. *IEEE Transactions on Information Theory*, 38(2):719–746, March 1992.

[149] Z. Xiong, N.P. Galatsanos, and M.T. Orchard. Marginal Analysis Prioritization for Image Compression Based on a Hierarchical Wavelet Decomposition. volume 5, pages V–546–V–549, April 1993.

[150] M.J. Tsai, J.D. Villasenor, and F. Chen. Stack-Run Image Coding. *IEEE Transactions on Circuit and System for Video Technology*, 6(5):519–521, October 1996.

[151] V.R. Algazi and Jr. R.R. Estes. Analysis Based Coding of Image Transform and Subband Coefficients. *Proceeding of the SPIE: Applications of Digital Image Processing XVII*, 2564:11–21, 1995.

[152] M.J. Gormish, E.L. Schwartz, A. Keith, M. Boliek, and A. Zandi. Lossless and Nearly Lossless Compression for High Quality Images. In *Proceedings of SPIE*, Febuary 1997. http://www.crc.ricoh.com/CREW/.

[153] M. Boliek and M.J. Gormish. Next Generation Image Compression and Manipulation Using CREW. In *Proceedings ICIP-97 (IEEE International Conference on Image Processing)*, pages 567–570, Santa Barbara, October 1997.

[154] G.M. Davis. A Wavelet-Based Analysis of Fractal Image Compression. *IEEE Transactions on Image Processing*, 7(2):141–154, February 1998.

[155] C.D. Creusere. A New Method of Robust Image Compression Based on the Embedded Zerotree Wavelet Algorithm. *IEEE Transactions on Image Processing*, 6(10):1436–1446, October 1997.

[156] A. Said and W.A. Pearlman. A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. *IEEE Transactions on Circuit and System for Video Technology*, 6(3):243–250, June 1996.

[157] Q. Wang and M. Ghanbari. Scalable Coding of Very High Resolution Video Using the Virtual Zerotree. *IEEE Transactions on Circuit and System for Video Technology*, 7(5):719–727, October 1997.

[158] S.A. Martucci, I. Sodagar, T. Chiang, and T.-Q. Zhang. A Zerotree Wavelet Video Coder. *IEEE Transactions on Circuit and System for Video Technology*, 7(1):109–118, February 1997.

[159] W.A. Pearlman, B.-J. Kim, and Zixiang Xiong. *Wavelet Image and Video Compression*, chapter 24: Embedded Video Subband Coding with 3D SPIHT. Kluwer Academic Publisher, 1998.

[160] G. Karlsson and M. Vetterli. Three Dimentional Subband Coding of Video. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1100–1103, April 1988.

[161] D. Taubman and A. Zakhor. Multirate 3-D Subband Coding of Video. *IEEE Transactions on Image Processing*, 3(5):572–588, September 1994.

[162] J.-R. Ohm. Three-Dimensional Subband Coding with Motion Compensation. *IEEE Transactions on Image Processing*, 3(5):559–571, September 1994.

[163] A.S. Lewis and G. Knowles. Video Compression Using 3D Wavelet Transforms. *Electronics Letters*, 26(6):396–398, March 1990.

[164] J.-Y. Tham, S. Ranganath, and A.A. Kassim. Scalable Very Low Bit-rate Video Compression Using Motion Compensated 3-D Wavelet Decomposition. *IEEE ISPACS Workshop*, pages 38.7.1–38.7.5, November 1996.

[165] J.-Y. Tham, S. Ranganath, and A.A. Kassim. Highly Scalable Wavelet-Based Video Codec for Very Low Bit-rate Environment. *IEEE Journal on Selected Areas in Communications*, 16(1):12–27, January 1998. http://wavelets.math.nus.sg/thamjy/.

[166] B.-J. Kim and W.A. Pearlman. Embedded Wavelet Video Coder Using Three-Dimensional Set Partitioning in Hierarchical Tree (SPIHT). *IEEE Data Compression Conference Proceedings*, pages 251–260, March 1997.

[167] B.-J. Kim and W.A. Pearlman. Low-Delay Embedded 3-D Wavelet Color Video Coding with SPIHT. *Proc. SPIE 3309 Visual Communications and Image Processing'98*, January 1998.

[168] B.-J. Kim, Z. Xiong, and W.A. Pearlman. Low Bit-Rate Scalable Video Coding with 3-D Set Partition in Hierarchical Trees (3-D SPIHT). *IEEE Transactions on Circuit and System for Video Technology*, 10(8):1374–1387, December 2000. http://ipl.rpi.edu/publications/publications.html.

[169] V. Areekul and R.H. Bamberger. Directional Zerotree Image Coding. In *31st Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 684–688, Monterey, CA, November 1997.

[170] T.-H. Lan and A.H. Tewfik. MultiGrid Embedding (MGE) Image Coding. In *Proceedings of IEEE International Conference on Image Processing*, pages 369–373, Kobe, Japan, October 1999.

[171] P.N. Topiwala and T.D. Tran. Local Zerotree Coding. In *Proceedings of IEEE International Conference on Image Processing*, pages 279–282, Kobe, Japan, October 1999.

[172] A.C. Miguel, A.E. Mohr, and E.A. Riskin. SPIHT for Generalized Multiple Description Coding. In *Proceedings of IEEE International Conference on Image Processing*, pages 842–846, Kobe, Japan, October 1999.

[173] S. Iren and P.D. Amer. SPIHT-NC: Network-Conscious Zerotree. In *Data Compression Conference Proceedings*, pages 313–322, March 2000.

[174] F.W. Wheeler and W.A. Pearlman. Low-Memory Packetized SPIHT Image Compression. In *33rd Asilomar Conference on Signals, Systems and Computers*, pages 1193–1197, Monterey, CA, 1999.

[175] F.W. Wheeler and W.A. Pearlman. SPIHT Image Compression Without List. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages 2047–2050, June 2000.

[176] C-Y Su and B-F Wu. Image Coding Based on Embedded Recursive Zerotree. In *ISMIP '97*, pages 387–392, Taipei, Taiwan, December 1997.

[177] R.J. Safranek and J.D. Johnston. A Perceptually Tuned Sub-band Image Coder with Image Dependent Quantization and Post-quantization Data Compression. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 1945–1948, Glasgow, Scotland, May 1989.

[178] M. Nadenau. *Integration of Human Color Vision Models Into High Quality Image Compression*. PhD thesis, EPFL, Lausanne, Switzerland, 2000. http://dewww.epfl.ch/ nadenau/Research/publication.htm.

[179] E.P. Simoncelli, W.T. Freeman, E.H. Adelson, and D.J. Heeger. Shiftable Multiscale Transform. *IEEE Transactions on Information Theory*, 38:587–607, 1992.

[180] D. le Gall and A. Tabatabai. Sub-band Coding of Digital Images Using Symmetric Short Kernel Filters and Arithmetic Coding Techniques. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 761–764, April 1988.

[181] A.R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Lossless Image Compression Using Integer to Integer Wavelet Transforms. In *Proceedings of IEEE International Conference on Image Processing*, pages 596–599, 1997.

[182] M.D. Adams and F. Kossentini. Reversible Integer-to-Integer Wavelet Transforms for Image Compression: Performance Evaluation and Analysis. *IEEE Transactions on Image Processing*, 9(6):1010–1024, June 2000.

[183] ISO/IEC JTC 1/SC 29/WG 1. ISO/IEC FCD 15444-1: Information Technology - JPEG 2000 Image Coding System, March 2000. http://www.jpeg.org/FCD15444-1.htm.

[184] C. Christopoulos, A. Skodras, and T. Ebrahimi. The JPEG2000 Still Image Coding System: An Overview. *IEEE Transactions on Consumer Electronics*, 46(4):1103–1127, November 2000.

[185] D. Santa-Cruz, T. Ebrahimi, J. Askelöf, M. Larsson, and C.A. Christopoulos. JPEG 2000 Still Image Coding Versus Other Standards. In *Proceedings of the SPIE's 45th annual meeting, Applications of Digital Image Processing*, volume 4115, San Diego, CA, July 2000.

[186] S.-J. Choi and J.W. Woods. Motion-Compensated 3-D Subband Coding of Video. *IEEE Transactions on Image Processing*, 8(2):155–167, February 1999.

[187] Recommendation H.263. *Video Coding for Low Bit Rate Communication.* ITU, 1998.

[188] M.-C. Lee, W.-G. Chen, C.-L. B. Lin, C.Gu, T. Markoc, S.I. Zabinsky, and R. Szeliski. A Layered Video Object Coding System Using Sprite and Affine Motion Model. *IEEE Transactions on Circuit and System for Video Technology*, 7(1):130–145, February 1997.

[189] R. Talluri, K. Oehler, T. Bannon, J.D. Courtney, A. Das, and J. Liao. A Robust, Scalable, Object-Based Video Compression Technique for Low Bit-Rate Coding. *IEEE Transactions on Circuit and System for Video Technology*, 7(1):221–233, February 1997.

[190] M. Kim, J.G. Choi, D. Kim, H. Lee, M.H. Lee, C. Ahn, and Y.-S. Ho. A VOP Generation Tool: Automatic Segmentation of Moving Objects in Image Sequences Based on Spatio-Temporal Information. *IEEE Transactions on Circuit and System for Video Technology*, 9(8):1216–1226, December 1999.

[191] *The Digital Signal Processing Handbook*, chapter Morphological Signal and Image Processing, pages (74–1)–(74–30). CRC Press and IEEE Press, 1998.

[192] R.M. Haralick and L.G. Shapiro. *Computer and Robot Vision*, volume I, chapter Mathematical Morphology, pages 156–188. Addison-Wesley, 1993.

[193] N. Habili, A. Moini, and N. Burgess. Automatic Thresholding for Change Detection in Digital Video. In *Proceedings of VCIP 2000*, Perth, Australia, June 2000.

[194] F.G. Meyer, A.Z. Averbuch, and J.-O. Stromberg. Fast Adaptive Wavelet Packet Image Compression. *IEEE Transactions on Image Processing*, 9(5):792–800, May 2000.