CONTEXT-BASED BIT PLANE GOLOMB CODER FOR SCALABLE IMAGE CODING

ZHANG RONG

(B.E. (Hons.) USTC, PRC)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2005

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my supervisors, Prof. Lawrence Wong and Dr. Qibin Sun, for their constant guidance, encouragement and support during my graduate studies. Their knowledge, insight and kindness provided me lots of benefits.

I want to take this opportunity to thank Yu Rongshan for his thoughtful comments, academic advices and encouragement on my research. I have also benefited a lot from intersections with He Dajun, Zhou Zhicheng, Zhang Zhishou, Ye Shuiming, Li Zhi, researchers in the Pervasive Media Lab. Their valuable suggestions on my research and thesis are highly appreciated. Special thanks to Tran Quoc Long and Jia Yuting for the valuable discussions and help on both my courses and research. I also want to thank my officemates Lao Weilun, Wang Yang and Moritz Häberle for their friendship and support on my studies. In addition, I would like to thank my friends Zhu Xinglei, Li Rui and Niu Zhengyu for their friendship and help on my studies and daily life.

I am so grateful to Wei Zhang, my husband, for his love and encouragement during our years. His broad knowledge on engineering and computer science helps me a lot in my research, and his love encourages me to pursue my dreams. I also want to thank my parents for their love and years of nurturing and supporting my education. Thank Mum for her care, her guidance towards my studies. And thank Dad for his constant encouragement during my life.

LIST OF PUBLICATIONS

1. Rong Zhang, Rongshan Yu, Qibin Sun, Wai-Choong Wong, “A new bit-plane entropy coder for scalable image coding”, *IEEE Int. Conf. Multimedia & Expo*, 2005.
2. Rong Zhang, Qibin Sun, Wai-Choong Wong, “A BPGC-based scalable image entropy coder resilient to errors”, *IEEE Int. Conf. Image Processing*, 2005.
3. Rong Zhang, Qibin Sun, Wai-Choong Wong, “An efficient context based BPGC scalable image coder”, *IEEE Trans. on Circuit and Systems II*, (submitted).

TABLE OF CONTENTS

SUMMARY

With the increasing use of digital images and delivering those images over networks, scalable image compression becomes a very important technique. It not only saves storage space and network transmission bandwidth, but also provides rich functionalities such as resolution scalability, fidelity scalability and progressive transmission. Wavelet based image coding schemes such as the state-of-the-art image compression standard JPEG2000 are very attractive for scalable image coding.

In this thesis, we present the proposed wavelet-based coder, Context-based Bit Plane Golomb Coding (CB-BPGC) for scalable image coding. The basic idea of CB-BPGC is to combine Bit Plane Golomb Coding (BPGC), a low complexity embedded compression strategy for Laplacian distributed sources such as wavelet coefficients in HL, LH and HH subbands, with image context modeling techniques. Compared to the standard JPEG2000, CB-BPGC provides better lossless compression ratio and comparable lossy coding performance by exploring the characteristics of the wavelet coefficients. Fortunately, compression performance improvement is achieved together with lower complexity in CB-BPGC compared to JPEG2000.

The error resilience performance of CB-BPGC is also evaluated in this thesis. Compared to JPEG2000, CB-BPGC is more resilient to channel errors when simulated on the wireless Rayleigh fading channel. Both the Peak Signal-to-Noise

Ratio (PSNR) and the subjective performance of the corrupted images are better than those of JPEG2000.

LIST OF TABLES

LIST OF FIGURES

Chapter 1. INTRODUCTION

With the expanding use of modern multimedia applications, the number of digital images is growing rapidly. Since the data used to represent images can be very large, image compression is one of the indispensable techniques to deal with the expansion of image data. Aiming to represent the images using as few bits as possible while satisfying certain quality requirement, image compression plays an important role in saving channel bandwidth in communication and also storage space for digital image data.

1.1. Background

Image compression has been a popular research topic for many years. The two fundamental components of image compression are *redundancy* reduction and *irrelevancy* reduction. Redundancy reduction refers to removing the statistical correlations of the source, by which the original signals can be exactly reconstructed; irrelevancy reduction aims to omit less important parts of the signal, by which the reconstructed signal is not exactly the original one but without bringing visual loss.

1.1.1. A general image compression system

A general image encoding and decoding system is illustrated in Figure 1-1. As shown in the figure, the encoding part includes three closely connected

components, the *transform*, the *quantizer* and the *encoder* while the decoding part consists of the inverse ones, the *decoder*, the *dequantizer* and the *inverse transform*.

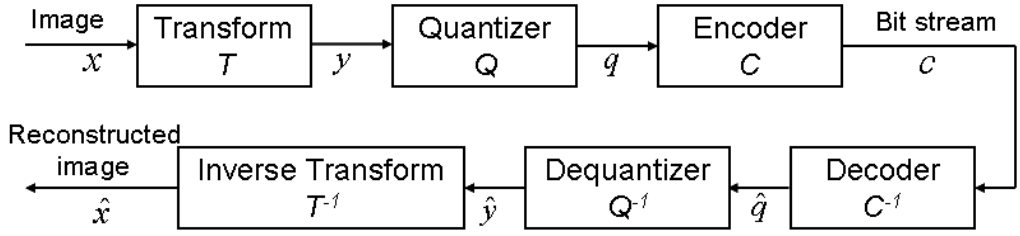


Figure 1-1 Block diagram of image compression system

Generally, images are never directly raw bits compressed by coding algorithms and image coding is much more than general purpose compression methods. This is because in most images, which are always represented by a two-dimensional array of intensity values, the intensity values of the neighboring pixels are heavily correlated. The transform in the image compression system is applied to remove these correlations. It can be Linear Prediction, Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT) or others, each with its own advantages and disadvantages. After the transformation, the transformed data which is more compressible is further quantized into a finite set of values. Finally, the entropy coder is applied to remove the redundancy of the quantized data. The decoding part of the image compression system is the inverse process of the encoding part. It is usually of lower complexity and performs faster than the encoding part.

According to the reconstructed images, image compression schemes can be classified into two types, *lossless coding* and *lossy coding*. Lossless coding

methods encode the images only by redundancy reduction where we can reconstruct exactly the same images as the original ones, but with a moderate compression performance. Lossy coding schemes, which use both redundancy and irrelevancy reduction techniques, achieve much higher compression while suffering some image quality degradation compared to the original images. However, if the lossy coding algorithms do not target at very high compression ratio, reconstructed images with no significantly visible loss can be achieved, which is also called *perceptual lossless coding*.

1.1.2. Image transmission over noisy channels

As more and more multimedia sources are distributed over the Internet and wireless mobile networks, robust transmission of these compressed data has become an increasingly important requirement since these channels are error-prone. Figure 1-2 shows the process of image encoding, decoding and transmission over adverse channels. The challenge of robust transmission is to protect the compressed data against adverse channel conditions while reducing the impact on bandwidth efficiency, a process called *error resilient* techniques.

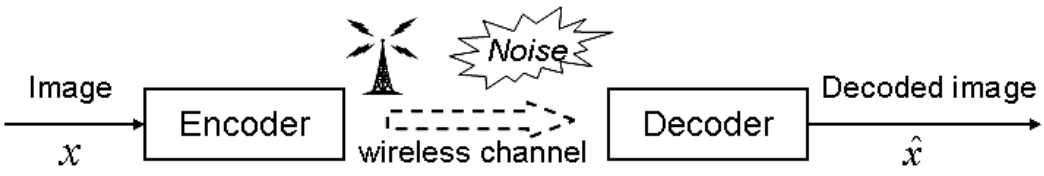


Figure 1-2 Image encoding, decoding and transmission over noisy channels

The error resilient techniques can be set up at the source coding level, the

channel coding level or both. Resynchronizaiton tools, such as segmentation and packetization of the bitstreams are often used to ensure independent decoding of the coruppted data and thus prevent error propagation. Self-recovery coding algorithms can also be included, such as reversible various length codes (RVLC), with which we can apply backward decoding to continue reconstructing the images when error is detected in the forward decoding process.

Additionally, channel coding techniques such as forward error correction (FEC) can be used to detect and further possibly correct errors without requesting retransimission of the original bitstreams. In some applications, if retransmission is possible, automatic repeat request (ARQ) protocols can be used to request retransimission of the lost data.

Except for the above techniques which are responsible for protecting the bitstream against noise, there are also some other error recovery ways, such as error concealment based on interpolation or edge filter methods to conceal errors in the damaged images in a post processing way.

1.2. Motivation and objective

With the ever-growing requirements from various applications, compression ratio is no longer the only concern in image coding. Some other features such as low computational complexity, resolution scalability, distortion scalability, region of interest, random access, and error resilience are also required by some applications. The international image compression standard JPEG2000, which

applies several state-of-the-art techniques, specifies such an attractive image coder which provides not only superior rate-distortion, subjective image quality but also rich functionalities.

However, behind the attractive features of JPEG2000 is the increase in computational complexity. As lower complexity coder is more practical than the increase in compression ratio for some applications [5], it is desirable to develop certain new image coders which achieve comparable coding performance as the current standard and provide rich functionalities but have lower complexity.

Based on an efficient and low complexity coding scheme, Bit Plane Golomb Coding (BPGC) developed for Laplacian distributed signals which is now successfully applied in scalable audio coding, we study the feasibility of this algorithm in scalable image coding. By exploring the distribution characteristics of the wavelet coefficients in the coding algorithm, we aim to develop a new image entropy coder which provides comparative coding performance and also rich features as the standard JPEG2000 but with lower complexity. Additionally, we also intend to improve the error resilience performance of the new image coder compared to that of JPEG2000 operating in a wireless Rayleigh fading channel

1.3. Organization of the thesis

This thesis is organized as follows. We briefly review some related techniques in wavelet based scalable image coding in Chapter 2, such as wavelet transform, quantization, bit plane coding, entropy coding and some well-known scalable

image coding examples.

In Chapter 3, we first review the embedded coding strategy, BPGC and then introduce the proposed Context-based Bit-Plane Golomb Coding (CB-BPGC) for scalable image coding. Comparison of both the PSNR and visually subjective performance between the proposed coder and the standard JPEG2000 are presented in this chapter. We also include a complexity analysis of CB-BPGC at the end of this chapter.

A brief review of error resilience techniques is given in Chapter 4, followed by the error resilience strategies used in CB-BPGC. In this chapter, we also show the experimental results of the error resilience performance of the two coders.

Chapter 5 then gives the concluding remarks of this thesis.

Chapter 2. WAVELET-BASED SCALABLE IMAGE CODING

As the requirement of progressive image transmission over the Internet and mobile networks increases, scalability becomes a more and more important feature for image compression systems. Wavelet based image coding algorithm has received lots of attention in image compression because it provides great potential to support scalability requirements [1][2][3][4][6].

In this chapter, firstly, we briefly review the general components in the wavelet based image coding systems, for example, wavelet transform, quantization techniques and entropy coding algorithms like arithmetic coder. Some successful scalable image coding examples such as the embedded zerotree wavelet coding (EZW) [1], the set partitioning in hierarchical trees (SPIHT) [2] and the embedded block coding with optimal truncation (EBCOT) [6] are introduced. We also briefly review the state-of-the-art JPEG2000 image coding standard [8].

2.1. Scalability

Scalability is a desirable requirement in multimedia encoding since:

- It is difficult for the encoder to encode the multimedia data and then save the compressed files for every bitrate due to storage and computation time constraints.
- In transmission, different clients may have different bitrate demands or

different transmission bandwidths, but the encoder has no idea to which client this compressed data will be sent and does not know which bitrate should be used in the encoding process.

- Even for a given client, the data transmission rate may be occasionally changed because of network condition changes such as fluctuations of channel bandwidth.

So, we need scalable coding to provide a single bitstream which can satisfy client demands and network condition changes. Bitstreams of various bitrates can be extracted from that single bitstream while partially discarding some bits to obtain a coarse but efficient representation or a lower resolution image. Once the image data is compressed, it can be decompressed in different ways depending on how much information is extracted from that single bitstream [7].

Generally, resolution (spatial) scalability and distortion (SNR or fidelity) scalability are the main scalability features in image compression. Resolution scalability aims to create bitstreams with distinct subsets of successive resolution levels. Distortion scalability refers to creating bitstreams with distinct subsets that successively refine the image quality (reducing the distortion) [7].

Wavelet-based image coding algorithms are very popular in designing scalable image coding systems because of the attractive feature of the wavelet transform. Wavelet transform is a tree-structured multi-resolution subband transform, which not only compacts most of the image energy into only a few low frequency subbands coefficients to make the data more compressible, but also makes the

decoding of resolution scalable bitstreams possible [23]. We briefly review wavelet transform in the next section.

2.2. Wavelet transform

Similar to transforms such as Fourier Transform, the wavelet transform is a time-frequency analysis tool which analyzes a signal's frequency content at a certain time point. However, wavelet analysis provides an alternative way to the traditional Fourier analysis for localizing both the time and frequency components in the time-frequency analysis [21].

Although Fourier transforms are very powerful in some of the signal processing fields, they also have some limitations. It is well-known that there is a tradeoff between the control of time and frequency resolution in the time-frequency analysis process, i.e., the finer the time resolution of the analysis, the more coarse the frequency resolution of the analysis. As a result, some applications which emphasize a finer frequency resolution will suffer from poor time localization and thus fail to isolate transients of the input signals [23].

Wavelet analysis then remedies these drawbacks of Fourier transforms. A comparison of the time-frequency planes of the Short Time Fourier Transform (STFT) and the Discrete Wavelet Transform (DWT) is given in Figure 2-1. As indicated in the figure, STFT has a uniform division of the frequency and time components throughout the time-frequency plane while DWT divides the time-frequency plane in a different, non-uniform manner [20].

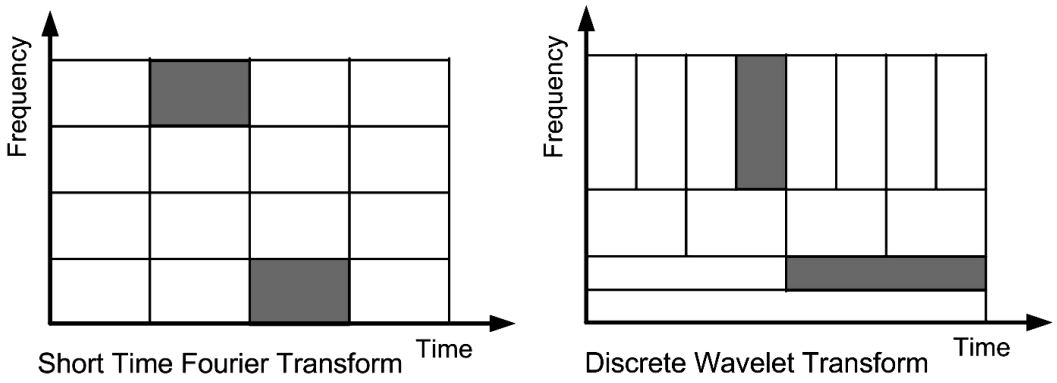


Figure 2-1 Comparison of time-frequency analysis of STFT (left) and DWT (right), each rectangle in the graphics represents a transform coefficient.

Generally, wavelet analysis provides finer frequency resolution at low frequencies and finer time resolution at high frequencies. That is often beneficial because the lower frequency components, which usually carry the main features of the signal, are distinguished from each other in terms of frequency contents. The wider temporal window also makes these features more global. For the higher frequency components, the temporal resolution is higher, from which we can capture the more detailed changes of the input signals.

In Figure 2-1, each rectangle has a corresponding transform coefficient and is related to a transform basis function. For the STFT, each basis function $\varphi_{(s,t)}(x)$ is the translation *t* and/or scaling *s* of a sinusoid waveform which is non-local and stretches out to infinity as shown in Figure 2-2 .

$$\varphi(x) = \sin(x), \qquad \varphi_{(s,t)}(x) = \sin(sx - t) \tag{2.1}$$

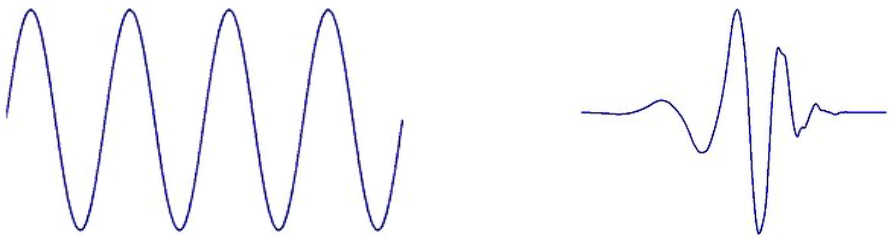


Figure 2-2 Comparison of sine wave (left) and Daubechies_10 wavelet (right)

For the DWT, each basis function $\phi_{(s,t)}(x)$ is the translation t and/or scaling s (usually powers of two) of a single shape which is called the mother wavelet.

$$\phi_{(s,t)}(x) = 2^{-\frac{s}{2}}\phi(2^{-s}x - t) \tag{2.2}$$

There may be different kinds of shapes for mother wavelets depending on the specific applications [23]. Figure 2-2 gives an example of the Daubechies_10 mother wavelet of the Daubechies wavelet family which is irregular in shape and compactly supported compared to the sine wave. It is these irregularities in shape and compactly supported properties that make wavelets an ideal tool for analyzing non-stationary signals. The irregular shape lends to analyzing signals with discontinuities or sharp changes, while the compactly supported nature makes for temporal localization of signal features [21].

Wavelet transform is now widely used in many applications such as denoising signals, musical tones analysis, and feature extraction. One of the most popular applications of wavelet analysis is image compression. The JPEG2000 standard, which is designed to update and replace the current JPEG standard, uses wavelet transform instead of Discrete Cosine Transform (DCT), to perform decomposition of images.

Usually, the two-dimensional decomposition of images is conducted by one-dimensional filters on the columns first and then on the rows separately [22]. As shown in Figure 2-3, an $N \times M$ image is decomposed by two successive steps of one-dimensional wavelet transform. We filter each column and then downsample to obtain two $N/2 \times M$ sub images. We then filter each row and downsample the

output to obtain four $N/2 \times M/2$ sub images. The "LL" sub image refers to the one by low-pass filtering both the column and row data; the "HL" one is obtained by low-pass filtering the column data and high-pass filtering the row data; the one obtained by high-pass filtering the column data and low-pass filtering the row data is called "LH" sub image; and the "HH" refers to the one by high-pass filtering both the column and row data.

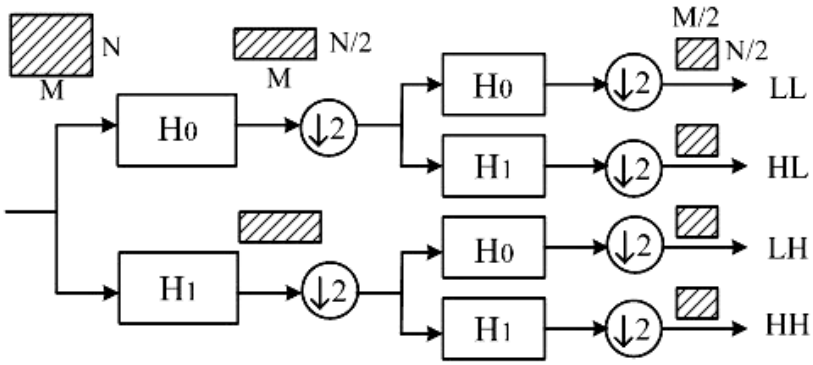
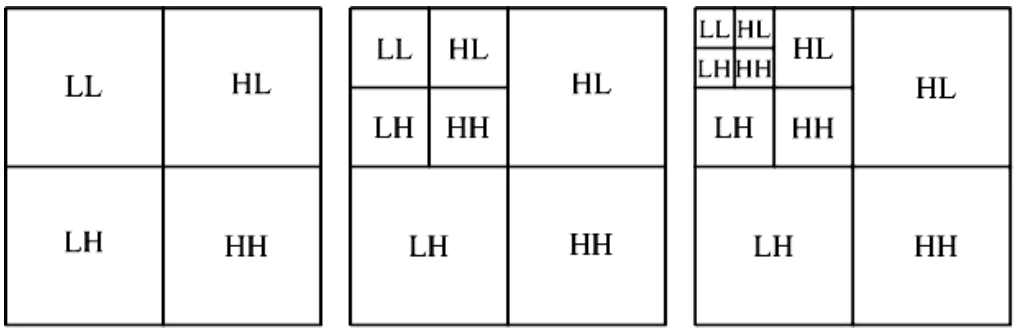


Figure 2-3 Wavelet decomposition of an $N \times M$ image, vertical filtering first and horizontal filtering second

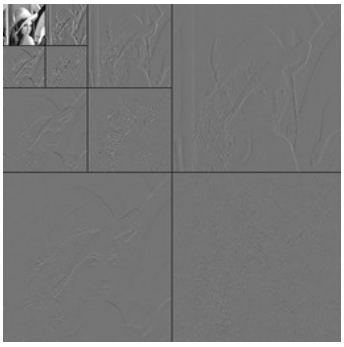
By recursively applying the wavelet decomposition as described above to the LL subband, a tree-structured wavelet transform with different levels of decomposition is obtained as illustrated in Figure 2-4. This multi-resolution property is particularly interesting for image compression applications since it provides for resolution scalability.



(a) (b) (c)

Figure 2-4 Wavelet decomposition (a) One level; (b) Two levels; (c) Three levels

(a)



(b)

Figure 2-5 (a) Image *lena* (512×512), (b) 3-level wavelet decomposition of image *lena* (the wavelet coefficients are shown in gray scale image, range [-127, 127])

An example of the 3-level wavelet decomposition of the image *lena* is shown in Figure 2-5. We can see from Figure 2-5 (b) that the wavelet transform highly compacts the energy, i.e., most of the wavelet coefficients with large magnitude localize in the higher level decomposition subbands, for example the LL band. Actually, the LL band is a low resolution version of the original image, which contains the general features of the original image. The coefficients in other subbands carry the more detailed information of the image, such as edge information. The HL bands also most strongly respond to vertical edges; the LH bands then contain mostly horizontal edges; and the HH bands correspond primarily to diagonally oriented details [7].

Unlike the traditional DCT based coders, where each coefficient corresponds to a fixed size spatial area and fixed frequency bandwidth and thus edge information disperse onto many non-zero coefficients, in order to achieve lower bitrate some edge information is lost and thus results in blocky artifacts. The wavelet multi-resolution representation ensures the major features (the lower frequency components) and the finer edge information of the original image occur in scales,

such that for low bitrate coding, there is no such blocky effect but only kind of blurring effect occurs, which is because of the discarding of coefficients in the high frequency subbands that are responsible for the finer detailed edge features.

2.3. Quantization

Generally, $N \times M$ images are represented by a two-dimensional integer array X with pixel elements $x[n,m]$. However, the transformed coefficients $y[n,m]$ are often no longer integers and a quantization step should be included before entropy coding. Quantization is often the only source of distortion in lossy compression that is responsible for reducing the precision of the signal and thus makes it much more compressible. While reducing the bits needed to represent the signal, it also brings loss of information, i.e., distortion. Thus, there is often no quantization process in lossless data compression.

2.3.1. Rate distortion theory

Rate distortion theory is concerned with the trade-off between rate and distortion in lossy compression schemes [22]. Rate is the average number of bits used to represent sample values. There are many approaches to measure the distortion of the reconstructed image. The most commonly used measurement is the Mean Square Error (MSE), defined by

$$MSE = \frac{1}{N \times M} \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} (x[n,m] - \hat{x}[n,m])^2 , \tag{2.3}$$

where $x[n,m]$ is the original pixel and $\hat{x}[n,m]$ is the reconstructed pixel. In image

compression, for an image sampled to fixed length B bits, the MSE is often expressed in an equivalent measure, Peak Signal-to-Noise Ratio (PSNR).

$$PSNR = 10\log_{10}\frac{(2^B - 1)^2}{MSE} \tag{2.4}$$

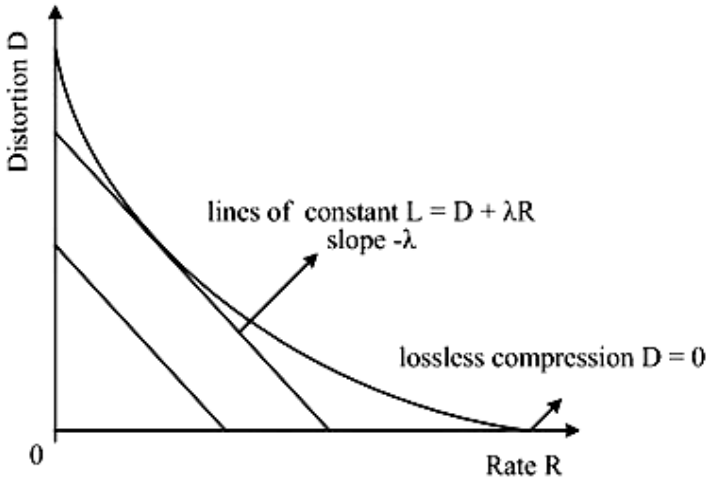
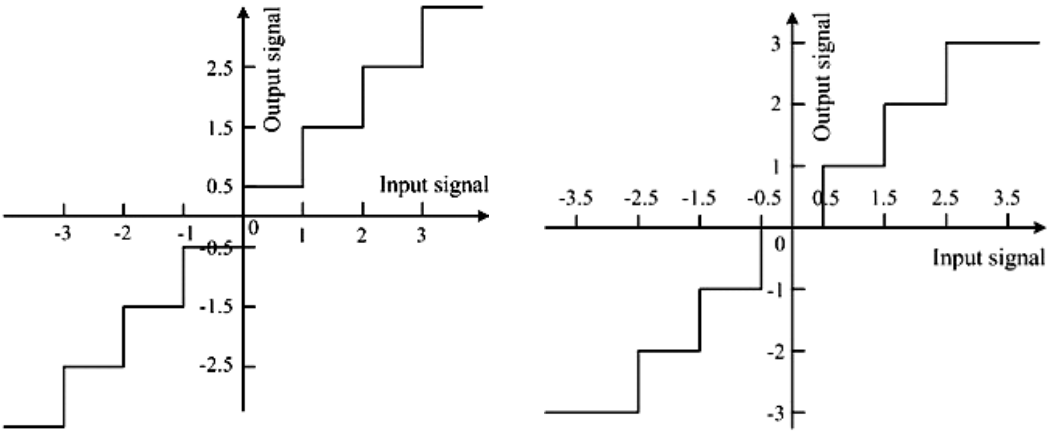


Figure 2-6 Rate distortion curve

The rate distortion function *R(D)*, which is a way to represent the trade-off between rate and distortion, specifies the lowest rate at which the source data can be encoded while satisfying the distortion less than or equal to a value *D*. Figure 2-6 gives an example of the rate distortion curve. Generally, the higher the bitrate, the smaller the distortion. When the distortion $D = 0$, the image is losslessly compressed. The Lagrangian cost function $L = D+\lambda R$ can be used to solve the minimization distortion under certain constrained rate problems.

The rate distortion theory is often used for solving problems of bit allocation in compression. Depending on the importance of the information it contains, each set of data is allocated a portion of the total bit budget while keeping the compressed image within a minimum possible distortion.



(a) (b)

Figure 2-7 (a) A midrise quantizer; (b) A midtread quantizer

2.3.2. Scalar quantization

The process of representing a large set of values (possibly infinite) with a much smaller set while bringing certain fidelity loss is called *quantization* [22]. According to the sets of quantizer input, it can be classified into *scalar quantization* (SQ) in which each quantizer output represents a single input data, and *vector quantization* (VQ) where the quantizer operates on blocks of data and the output represents a bunch of input samples.

The scalar quantizer is quite simple. Figure 2-7 gives examples of the scalar midrise quantizer and the midtread quantizer. Both of them are uniform quantizers where each input sample is represented by the middle value in the interval with a quantization step size $\Delta = 1$, but the midtread quantizer has zero as one of its levels while the midrise one does not have.

It is especially useful for the midtread quantizer in situations where it is important to represent a zero value, for example, in audio processing zeros are

needed to represent silent periods. Note that the midtread quantizer has an odd number of quantization levels while midrise quantizer has an even number. That means if a fixed length 3-bit code is used, we have eight levels for the midrise quantizer and seven levels for the midtread one, where one codeword is wasted.

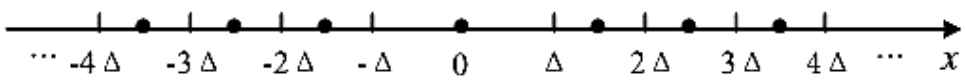


Figure 2-8 Uniform scalar quantization with a 2Δ wide dead-zone

Usually, for sources with zero mean, a small improvement of the rate-distortion function *R(D)* can be obtained by widening the midtread zero value interval, which is often called the *dead-zone*. A uniform SQ with a 2Δ wide dead-zone is illustrated in Figure 2-8 (Δ is the quantization step size). This quantizer can be implemented as

$$q = Q(x) = \begin{cases} sign(x)\left\lfloor \dfrac{|x|}{\Delta} \right\rfloor & |x| > \Delta \\ 0 & otherwise \end{cases}. \tag{2.5}$$

And the corresponding dequantizer is defined as

$$\hat{x}_q = \begin{cases} sign(q)|q|\Delta & q \neq 0 \\ 0 & q = 0 \end{cases}. \tag{2.6}$$

Uniform SQ is one of the simplest quantization schemes. SQ can also be non-uniform and designed to optimally adapt to the signal's *probability density function* (*pdf*). On the other hand, VQ represents a bunch of input samples by a codeword but have a much higher computational complexity. We will not discuss the details of these VQ techniques. For these detailed descriptions, please refer to [22].

2.4. Bit plane coding

As mentioned in Section 2.1, a very desirable feature of a compression system is the ability to successively refine the reconstructed data as the bitstream is decoded, i.e., the ability of scalability. Embedded coding is the key technique to achieve distortion scalability. The main advantage of the embedded bitstream lies in its ability to generate a compression bitstream which can be dynamically truncated to fit a certain rate, distortion or complexity constrains without loss of optimality.

Table 2-1 An example of bit plane coding
Sample data range: [-63, 63], the most significant bit plane: $m = 5$

Samples	Value	Bit Planes B_j ($j = m,m\text{-}1,\ldots,0$)						
		Sign	$j = 5$	$j = 4$	$j = 3$	$j = 2$	$j = 1$	$j = 0$
x_0	34	+	1	0	0	0	1	0
x_1	-6	-	0	0	0	1	1	0
x_2	3	+	0	0	0	0	1	1
x_3	23	+	0	1	0	1	1	1
x_4	-52	-	1	1	0	1	0	0
x_5	49	+	1	1	0	0	0	1
x_6	-11	-	0	0	1	0	1	1
...	...	...	...	...	...	...	...	...

Bit plane coding (BPC) is then a natural and simple approach to implement an embedded coding system. It is included in most of the embedded image, audio and video coding systems [1][2][3][4][6][16][26]. The general idea of BPC is quite simple. The input data are first represented in magnitude and sign parts; the magnitude part is then binary represented as shown in Table 2-1. A set of data range in [-63, 63] has 6 bit planes, from the most significant 5^{th} bit plane to the least significant 0^{th} bit plane. It is then sequentially coded by bit planes, normally from the most significant bit plane to the least significant one to successively

refine the bitstreams.

In some embedded image coding systems, such as Embedded Block Coding with Optimal Truncation (EBCOT) in [6] and Pixel Classification and Sorting (PCAS) in [16], a code block is often encoded bit plane by bit plane in a certain order, e.g. raster order. And in order to obtain fine granular scalability, they operate on fractional bit planes where the BPC process often includes significant coding pass and magnitude refinement coding pass. Some other schemes such as Rate-Distortion optimized Embedding (RDE) introduced in [4] encode bits not in bit plane sequential order but encode several bit planes together according to the expected R-D slopes. In that method, when not all the bits in the 5^{th} bit plane have already been encoded, some bits in the 4^{th} bit plane are going to be encoded. We will further discuss the different bit plane coding techniques used in different coding examples in Section 2.6.

2.5. Entropy coding

After the transformed coefficients have been quantized to a finite set of values, they are often first operated by some source modeling methods. The modeling methods are responsible for gathering statistics and identifying data contexts which make the source models more accurate and reliable. They are then followed by an *entropy coding* process.

Entropy coding refers to representation of the input data in the most compact form. It may be responsible for almost all the compression effort, or it just gives

some additional compression as a complement to the previous processing stages.

2.5.1. Entropy and compression

Entropy in information theory means how much randomness is in a signal or alternatively how much information is carried by the signal [17]. Given the probability p of a discrete random variable X which has n states, entropy is formally defined by

$$H(x) = -\sum_{i=1}^{n} p(i) \log_2 p(i) . \tag{2.7}$$

Entropy can measure information in units of bits. It provides fundamental bounds on coding performance. Shannon points out in [17] that the entropy rate of a random process provides a lower bound on the average number of bits which must be spent in coding and also that this bound may be approached arbitrarily closely as the complexity of the coding scheme is allowed to grow without bound.

Most of the entropy coding methods fall into two classes: *dictionary* based schemes and *statistical* schemes. Dictionary based compression algorithms operate by replacing groups of symbols in the input text with fixed length codes, e.g. the well known Lempel-Zif-Welch (LZW) algorithm [22]. Statistical entropy coding methods operate by encoding symbols into variable length codes and the length of the codes varies according to the probability of the symbol. Symbols with a lower probability are encoded by more bits, while higher frequency symbols are encoded by fewer bits.

2.5.2. Arithmetic coding

Among all the entropy coding methods, a statistical entropy coding scheme, *arithmetic coding* stands out for its elegance, effectiveness, and versatility [24]. It is widely used in compression algorithms such as JPEG2000 [8], MPEG-4 Scalable Audio Coding standard [26] and video coding standard H.264.

When applied to *independent and identically distributed* (*i.i.d.*) sources, an arithmetic coder provides proven optimal compression. For those non i.i.d. sources, by combining with context modeling techniques it yields near-optimal or significantly improved compression. In addition, it is especially useful to deal with sources with small alphabets, such as binary sources, and alphabets with highly skewed probabilities.

In arithmetic coding, a sequence of symbols is represented by an interval of real numbers between 0 and 1. The *cumulative distribution function* (*cdf*) $F_x(i)$ is used to map the sequence into intervals. We are going to explain the idea behind arithmetic coding through an example.

Table 2-2 Example: fixed model for alphabet {a, e, o, !}

Symbols	Probability	Subintervals
a	0.2	[0, 0.2)
e	0.2	[0.2, 0.4)
o	0.4	[0.4, 0.8)
!	0.2	[0.8, 1)

Suppose we want to encode the sequence *eaoo!* with the probability distribution $P(x_i)$ (i=0, 1, 2, 3) listed in Table 2-2. The unit interval [0, 1) is divided to subintervals [$F_x(i\text{-}1)$, $F_x(i)$) with the symbol x_i. As illustrated in Figure 2-9, at the

beginning, the interval is [0, 1) and the first symbol, *e*, falls in the interval of [0.2, 0.4), therefore, after encoding, the lower limits $l^{(1)}$ of the new interval is 0.2 and the upper limits $u^{(1)}$ is 0.4. The next symbol to be encoded is *a*, with a range [0, 0.2) in the unit interval. Thus, after encoding the symbol *a*, the lower and the upper limits of the current interval are $l^{(2)} = 0.2$, $u^{(2)} = 0.24$. The updating of the interval can be written as follows,

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_{n-1}), \tag{2.8}$$

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n). \tag{2.9}$$

Applying the updating intervals for the whole sequence, we get the final interval [0.22752, 0.2288) to represent the sequence. This process is described graphically in Figure 2-9. The decoding then just mimics the encoding process to extract the original bit according to its probability and the current interval.

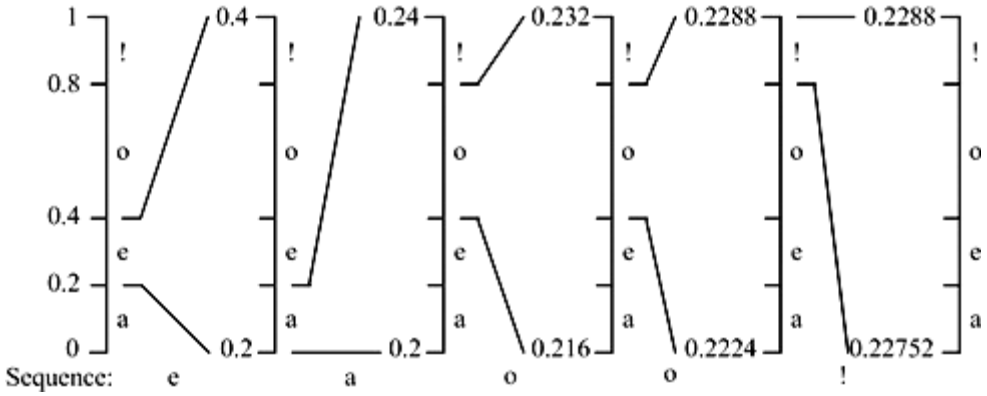


Figure 2-9 Representation of the arithmetic coding process with interval at each stage

Apparently, as the sequence becomes longer, the width of the interval can become smaller and smaller and sometimes it can be small enough to map different symbols onto the same interval which probably causes wrongly decoded symbols. That precision problem prohibited arithmetic coding from practical

usage for years and finally was solved in 1970s. Witten, et al [18] gave a detailed C implementation of the arithmetic coding.

In the encoding process, the probability model can be updated after each symbol is encoded, which is different from static arithmetic coding for applying a probability estimation procedure. Adaptive arithmetic coding receives lots of attention for its coding effectiveness, however, with a higher complexity [31]. Some other variants of the basic arithmetic coding algorithm also exist, such as the multiplication-free binary coder, *Q coder* [19] and the *MQ coder*, the binary adaptive arithmetic coder which is used in the image coding standards JBIG [9] and JPEG2000 [8].

2.6. Scalable image coding examples

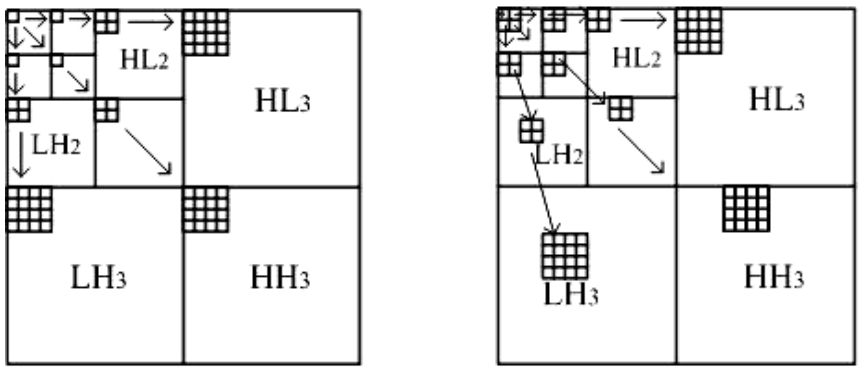
In the framework of embedded image coding system, the first stage is transform and quantization, the second stage is modeling and ordering, and the last stage is entropy coding and post processing [14]. Previous research works show that modeling and ordering are very important to design a successful embedded coder. Most of the wavelet based scalable image coding schemes gain compression effectiveness by exploring the interscale or intrascale wavelet coefficient correlations or both. In this section, we review some embedded image coding schemes.

2.6.1. EZW

The EZW algorithm was first presented in [1] by Shapiro, which became a

milestone for embedded image coding and produced the state-of-the-art compression performance at that time. It explores the so-called wavelet coefficients structure, *zerotrees* and achieves embedding via binary BPC.

Different from the raster scan of image bit planes or the progressively "zig zag" scan of the DCT coefficient bit planes, EZW encodes the larger magnitude coefficients bit planes first, which are supposed to contain the more important information of the original image, and allocates as few as possible bits to the near zero values. This is obtained from the structure "zerotrees", which means given a threshold *T*, if the current coefficient (*parent*) is smaller than *T*, then all of its corresponding spatial location coefficients in the higher frequency subbands (*children*) tend to be smaller than *T*, and we do not encode the bit planes of all coefficients in this zerotree now because they seem less important compared to the coefficients greater then *T*.



(a) (b)

Figure 2-10 (a) EZW parent-child relationship; (b) SPIHT parent-child relationship

The parent and child relationship in EZW is illustrated in Figure 2-10 (a). In general, a coefficient in subband HL_d, LH_d or HH_d has 4 children, 16 grandchildren, 64 great-grandchildren, etc. A coefficient in the LL_d has 3 children,

12 grandchildren, 48 great-grandchildren, etc.

The embedding bitstream is achieved by comparing the wavelet coefficient magnitudes to a set of octavely decreasing thresholds $T_k = T_0 2^{-k}$, where T_0 is chosen to satisfy $|y|_{max}/2 < T_0 < |y|_{max}$ ($|y|_{max}$ is the maximum magnitude for all coefficients). At the beginning, each insignificant coefficient, whose bit planes are not coded yet, is compared to T_0 in raster order, first within LL_D, then HL_D, LH_D, HH_D, then HL_{D-1}, and so on. Coding is accomplished via a 4-ary alphabet: *POS* (the significant positive coefficient), *NEG* (the significant negative coefficient), *ZTR* (the zerotree root, which indicates the current coefficient and its offspring are all less than T_0), *IZ* (the isolated zero, which means the current coefficient is less than T_0 but at least one of its offspring is larger than T_0). For those three highest frequency subband coefficients, which have no children, the *ZTR* and *IZ* symbols are replaced by the single symbol *Z*. As the process goes into the higher frequency subbands, these coefficients which are already in a zerotree are not coded again. This coding pass is called *dominant pass* which operates on the insignificant coefficients.

After that, the threshold is changed to T_1 and the encoder goes to the next bit plane. A *subordinate pass* is first carried out to encode the refinement bit plane of the coefficients already significant in the previous bit planes, followed by the second dominant pass. The processing continues alternating between dominant and subordinate passes and can stop at any time for certain rate/distortion constraint.

Context based arithmetic coding [18] is then used to losslessly compress the sequences resulting from the procedure discussed above. The arithmetic coder encodes the 4-ary symbols in the dominant pass and the refinement symbols in the subordinate pass directly and uses scaled down probability model adaptation [18].

The EZW technique not only had competitive compression performance compared to other high complexity compression techniques at that time, but also was fast in execution and produced an embedded bitstream.

2.6.2. SPIHT

The SPIHT algorithm proposed in [2] is an extension and improvement of the EZW algorithm and has been regarded as a benchmark in embedded image compression. Some features in SPIHT remain the same as with EZW. However, there are also several significant differences.

Firstly, the order of the significant and refinement coding passes is reversed. The parent-child relationship of the coefficients in LL band is changed as shown in Figure 2-10 (b), where one fourth of the coefficients in the LL band have no children while the remaining ones have four children each in the corresponding subbands. There are also two kinds of zerotrees in SPIHT, *type A* which consists of a root with all the offsprings less than the threshold but the root itself need not be less than the threshold, and *type B* which is similar to type *A* but do not include the children of the root, i.e., only the grandchildren, great-grandchildren, etc.

Unlike EZW, in SPIHT, there are three ordered lists: *LSC*, list of significant

coefficients containing the coordinates of all the significant coefficients; *LIS*, list of insignificant sets of coefficients including the coordinates of the roots of sets type *A* and type *B*; *LIC*, list of insignificant coefficients containing the coordinates of the remaining coefficients.

Assume each coefficient is represented by the sign $s[i,j]$ and the magnitude bit planes $q_k[i,j]$. The SPIHT algorithm is then operated as follows,

(0) Initiation

- $k = 0$, $LSC = \Phi$, LIC = {all coordinates $[i, j]$ of coefficients in LL}, LIS = {all coordinates $[i, j]$ of coefficients in LL that have children}. Set all entries of the *LIS* to type *A*.

(1) Significant pass

- For each $[i,j]$ in *LIC*: output $q_k[i,j]$. If $q_k[i,j] = 1$, output $s[i,j]$ and move $[i,j]$ to the *LSC*.
- For each $[i,j]$ in *LIS*:

  i. Output "0" if current coefficient is insignificant; otherwise "1".

  ii. If the above output is "1",

  Type *A*: changed to Type *B* and sent to the bottom of the *LIS*. The $q_k[i,j]$ bits of each child are coded (with any required sign bit). The child is sent to the end of *LIC* or *LSC*, as appropriate.

  Type *B*: deleted from the *LIS*, and each child is added to the end of the *LIS* as set of Type *A*.

(2) Refinement pass

- For each $[i,j]$ in *LSC*: output $q_k[i,j]$ excluding the coefficients added to the *LSC* in the most recent significant pass.

(3) Set $k = k+1$ and go to step (1).

The arithmetic coder is used as the entropy coder in SPIHT. Unlike in EZW, here only symbols from the significant passes are coded while the refinement bits are uncoded, i.e., SPIHT only codes the symbol "1" and "0" of the significant passes and even the sign bits are left uncoded.

The SPIHT algorithm provides better compression performance than the EZW algorithm at an even lower level of complexity. Many other famous embedded image compression systems are also motivated by the key principles of set partitioning and sorting by significance in SPIHT, such as the Set Partitioning Embedded Block (SPECK) [12][13] and the Embedded Zero Block Coding (EZBC) [15].

2.6.3. EBCOT

EBCOT, proposed by Taubman in [6], is an entropy coder which is carried out after the wavelet transform and quantization processes. Unlike the EZW and SPIHT algorithms which exploit both the interscale and the intrascale correlations in forms of zerotrees, EBCOT captures only the intrascale correlation. Each subband is partitioned into relatively small code blocks (e.g. 64×64 or 16×16) and these code blocks are encoded independently as shown in Figure 2-11.

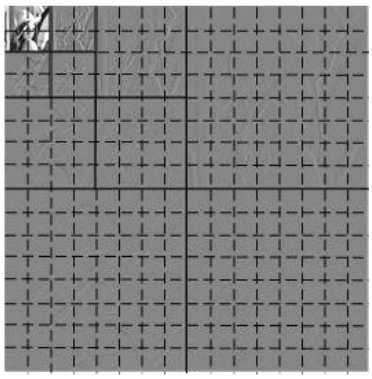


Figure 2-11 Partitioning image *lena* (256×256) to code blocks (16×16)

The disadvantage of independent block coding lies in that it is unable to explore redundancy between the blocks in the same subband and also the parent-child relationship in the higher and lower resolution corresponding subbands. However, because of the independent coding of blocks, EBCOT is able to embed resolution scalable bitstreams and is capable of random access and better error resilience. It also reduces the memory consumption in hardware implementations. In addition, the block coding in EBCOT also facilitates the ordering of the bitstreams by applying the post compression rate distortion optimization (PCRD) algorithm which we will discuss later.

The EBCOT algorithm is an independent block coded, context based adaptive bit plane coder, which is conceptually divided into two *Tiers* as shown in Figure 2-12. *Tier* 1 is the embedded block coding responsible for source modeling and entropy coding; while *Tier* 2 is the PCRD for ordering code block bitstreams in an optimal way to minimize the distortion subject to bitrate constraints and thus generating the output stream in packets.

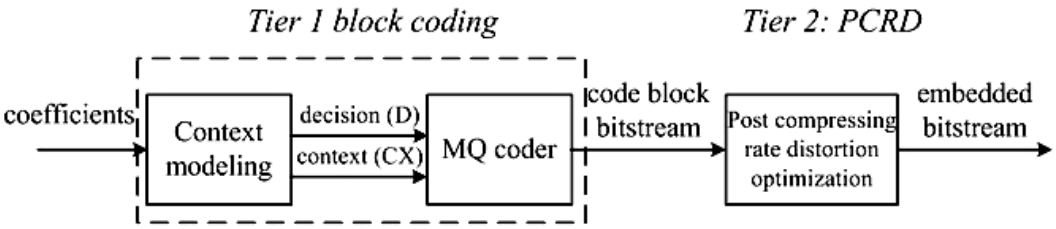


Figure 2-12 EBCOT *Tier 1* and *Tier 2*

More explicitly, in *Tier 1*, after coefficient subbands are divided into small code blocks, each code block is bit plane encoded. Each bit plane is scanned stripe by stripe and each stripe is scanned column by column as graphically shown in Figure 2-13. The bits in a certain bit plane are then coded by one of the three coding passes: significant propagation coding pass (SIG), magnitude refinement coding pass (MAR) and clear up coding pass (CLU).

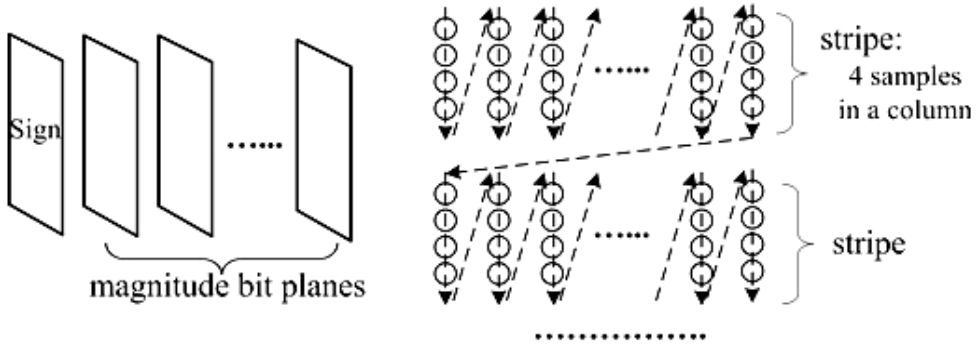


Figure 2-13 EBCOT bit plane coding and scanning order within a bit plane

Given a bit plane, the SIG coding pass encodes the bit whose corresponding coefficient is insignificant and has at least one neighbor (each coefficient has eight neighbors) already significant in the previous bit planes. These bits are the most likely to become significant and should be encoded earlier than the other bits in the current bit plane. If the bit is “1”, the sign coding should be followed and this coefficient is identified as significant in the processes of next bit planes. The MAR coding pass then refines the bits whose corresponding coefficients are

already significant. The remaining bits are then coded during the CLU coding pass. Obviously, each bit plane has these three coding passes except for the most significant bit plane which has only the CLU coding pass.

As we can see above, which pass a coefficient bit is coded in depends on the conditions or states of the corresponding coefficients. The coding passes give a fine partitioning of bit planes into three sets, providing more valid truncation points in the following PCRD optimization algorithms which will improve the embedded performance. In addition, four coding primitives are employed to obtain a finer source modeling: *Zero Coding* (ZC), *Sign Coding* (SC), *Magnitude Refinement* (MR) and *Run-length Coding* (RLC). The ZC and SC primitives are applied in the SIG coding pass; the MAR coding pass includes the MR primitive; and the CLU coding pass contains the ZC, SC and also the RLC primitives.

According to the different significant states of the eight neighbors, the ZC primitive has 9 contexts; the SC primitive has 5 contexts depending on the sign states of the horizontal and vertical four neighbors; the MR primitive then includes 3 contexts according to the significant states of the eight neighbors and whether this coefficient has been magnitude refined before; finally, the RLC primitive has only 1 context. So, there are in all 18 contexts modeled in EBCOT for all the three fractional coding passes. Each bit (binary decision) together with its context is then sent to the arithmetic coder. The arithmetic coder used in EBCOT is the adaptive binary coder, which is called the MQ coder.

After all the blocks are encoded, in *Tier* 2, the PCRD algorithm is applied. We

try to optimally select the truncation points, $\{n_i\}$ (with length L_i^{ni}, distortion D_i^n for code block B_i) so as to minimize the overall distortion, D, subject to an overall length constraint L_{max},

$$D = \sum D_i^{n_i}, \quad L_{\max} \geq L = \sum L_i^{n_i}. \tag{2.10}$$

The Lagrangian optimization can be used,

$$D(\lambda) + \lambda L(\lambda) = \sum (D_i^{(n_i,\lambda)} + \lambda L_i^{(n_i,\lambda)}). \tag{2.11}$$

The PCRD algorithm solves this problem by selecting the feasible truncation points which satisfy the convex hull property with decreasing D-L slopes which is shown in Figure 2-14.

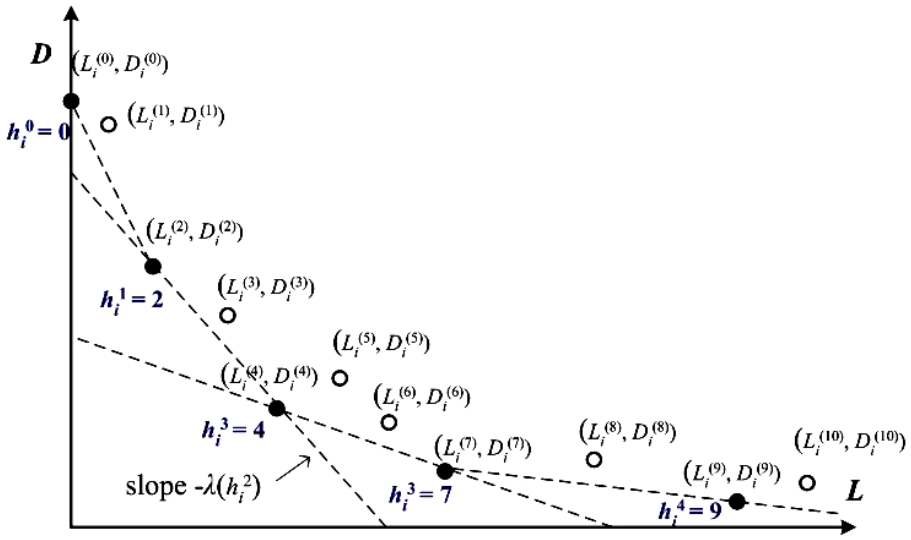


Figure 2-14 Convex hull formed by the feasible truncation points for block B_i

These feasible truncation points are candidates for the embedded bitstream truncation points. The EBCOT bitstream is finally organized in layers as shown in Figure 2-15. From the figure, we can see that, each code block has different contribution to a certain layer depending on its performance to reduce the distortion. Sometimes, this contribution is zero, which means there is no bitstream

from this block in the current layer.

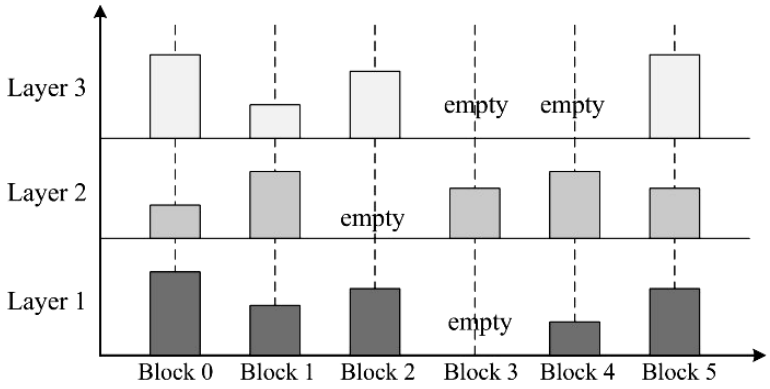


Figure 2-15 Code block contributions to quality layers (6 blocks and 3 layers)

The compression performance of EBCOT is better than the previous EZW and SPIHT algorithms [6]. In addition, EBCOT is a highly scalable compression algorithm together with attractive features like resolution scalability, SNR scalability and random access. It is selected to act as the entropy coder for the state-of-the-art image coding standard JPEG2000 [8].

2.7. JPEG2000

The block diagram of the image encoding, transmission and decoding in the image standard JPEG2000 is shown in Figure 2-16.

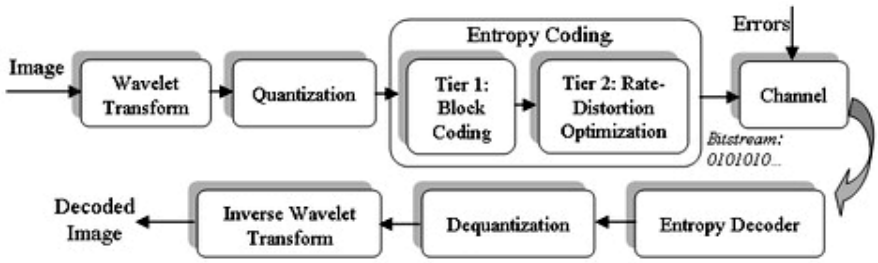


Figure 2-16 Image encoding, transmission and decoding of JPEG2000

In the standard, if the input is a color image, the first step is to apply the color transform, for example, from the RGB color space to the YCrCb space. Then each

color component is regarded as if they were grey scale images. They are then divided into blocks called *tiles*, which have disjoint codestream from each other.

The wavelet transform is then applied to the tiles. There are two types of discrete wavelet transform specified in JPEG2000, one is the reversible 5/3 LeGall filter and the other one is the irreversible Daubechies 9/7 filter. In lossy compression, a quantization step follows the wavelet transform. Two different quantization procedures are allowed: the dead-zone scalar quantization as discussed in Section 2.3.2 and Trellis-coded quantization. The entropy coder used in JPEG2000 is the EBCOT which is discussed in the last section.

The codestream of JPEG2000 is illustrated in Figure 2-17. Basically, JPEG2000 codestream is organized in packets, which contains a packet header and a packet body. The header includes some important parameter information and the body is the coded symbols.

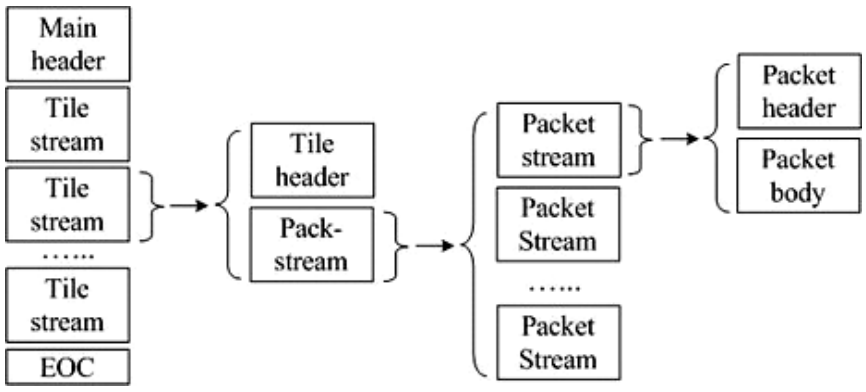


Figure 2-17 JPEG2000 code stream

JPEG2000 brings a new paradigm to the image compression [10]. It provides both lossy and lossless compression. A JPEG2000 codestream can be decompressed in many ways to obtain images with different resolutions and

fidelities. In addition to the resolution scalability and quality scalability, the JPEG2000 codestream also supports spatial random access. Each region can be accessed and decoded at a variety of resolutions and qualities. In addition, the JPEG2000 codestream also has the ability of error resilience when it is delivered over noise transmission channels.

Chapter 3. CONTEXT-BASED BIT PLANE GOLOMB CODING

We are going to present the proposed scalable image coder, Context-based Bit Plane Golomb Coding (CB-BPGC) in this chapter. It is motivated by the BPGC algorithm, an embedded coding scheme for Laplacian distributed sources which we assume are representative of wavelet coefficients in the HL, LH, and HH subbands, and the image context modeling techniques which explore the correlations between neighboring samples.

We will first discuss the BPGC algorithm and the context modeling techniques, followed by the detailed structure and implementation of the CB-BPGC coder for scalable image coding together with the evaluation of its compression performance compared to the JPEG2000 standard. A complexity analysis of the CB-BPGC algorithm is also included in this chapter.

3.1. Bit Plane Golomb Coding

The embedded coding strategy BPGC, which provides near optimal coding performance for sources with Laplacian distribution, was first presented in [25]. It is now successfully implemented in the latest MPEG-4 Audio Scalable Lossless Coding (SLS) Standard (also called AAZ coder) [26]. We start this section with a brief review of the algorithm, followed by a description of using BPGC in the AAZ audio coding and an analysis of the feasibility to use BPGC in scalable

image coding.

3.1.1. BPGC Algorithm

BPGC is one of the bit plane coding strategies which encodes the source symbols bit plane by bit plane as introduced in Section 2.4. However, BPGC is not a simple bit plane coder. It simplifies the bit plane coding of an independent and identically distributed (*i.i.d.*) Laplacian source by giving a static probability model for bits in each bit plane. These bits are then easily encoded by a static arithmetic coder whose input symbols include exactly the bit and the corresponding probability as discussed in Section 2.5.2.

Consider a Laplacian distributed source X, which has a *pdf* given by,

$$f_X(x) = e^{-|x|\sqrt{2/\sigma^2}} / \sqrt{2\sigma^2} \,. \tag{3.1}$$

Each sample x_i (i = 1, 2 …N) is binary represented by bit plane symbols $b_{i,j}$ (value 0 or 1) and the sign symbol s_i,

$$x_i = s_i \sum_j b_{i,j} 2^j, \qquad i = 1,...,N \qquad j = 0,...,m \tag{3.2}$$

$$s_i = \begin{cases} 1 & x_i \geq 0 \\ 0 & x_i < 0 \end{cases}, \tag{3.3}$$

where m is the most significant bit plane which satisfies:

$$2^m \leq \max\left\{|x_i|\right\} \leq 2^{m+1}. \tag{3.4}$$

If the source X is *i.i.d.*, the probability distributions of the bit plane symbol $b_{i,j}$ (value 0 and value 1) in the bit plane B_j can be written as,

$$prob(b_{i,j} = 1) \;=\; p_j \;=\; 1 - (1 + \theta^{2^j})^{-1} \tag{3.5}$$

and

$$prob(b_{i,j} = 0) \;=\; 1 - p_j \,, \tag{3.6}$$

where $$\theta = e^{-\sqrt{2/\sigma^2}} \tag{3.7}$$

is known as the distribution parameter which can be estimated from the statistical properties of the sample data, for example, the maximum likelihood (ML) estimation of θ is given by

$$\theta = e^{-N/A}, \tag{3.8}$$

where N is the number of the samples and A is the absolute sum of the samples.

From Equation (3.5), we can derive that the probability p_j using the following updating rule

$$p_j = \sqrt{p_{j+1}} / (\sqrt{1-p_{j+1}} + \sqrt{p_{j+1}}). \tag{3.9}$$

We can further simplify the probability of the bit $b_{i,j} = 1$, i.e. p_j in bit plane B_j (j = 0, 1, ... m) as follows [25]

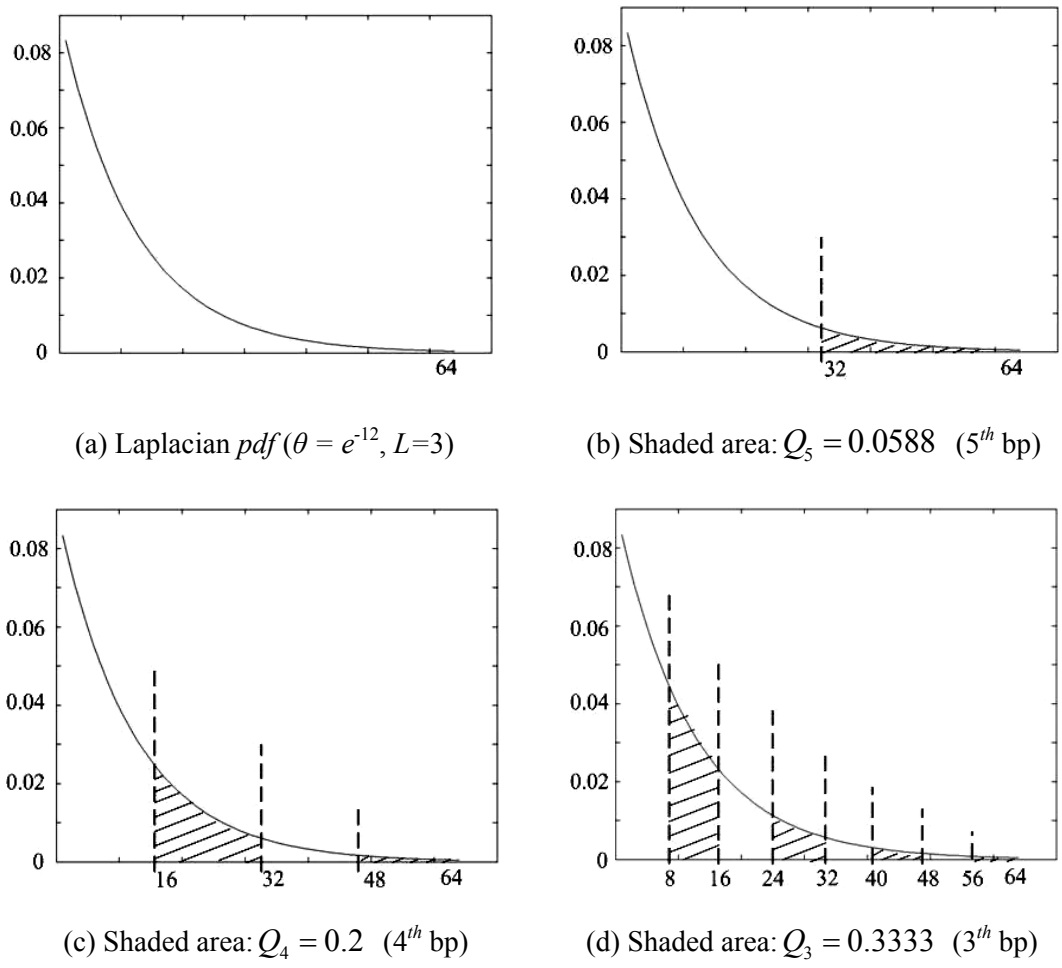
$$Q_j^L = \begin{cases} 1/\left(1+2^{2^{j-L}}\right) & j \geq L \\ 1/2 & j < L \end{cases} \tag{3.10}$$

$$L = \min\left\{L' \in Z \mid 2^{L'+1} N \geq A\right\}. \tag{3.11}$$

The approximate probability Q_j^L follows the probability updating rule of Equation (3.9) for bit planes from the most significant bit plane m to the bit plane L, and after the L^{th} bit plane, it enters to a so-called *lazy mode* where the bit probability is 1/2 for both bit value 1 and 0.

Therefore, the parameter L divides the bit planes into two parts: *lazy bit planes* (the $(L\text{-}1)^{th}$ bit plane to the 0^{th} bit plane) where bits 0 and 1 are uniformly distributed; and *non-lazy bit planes* (the m^{th} bit plane to the L^{th} bit plane) whose skew probabilities are specified by the distance from the current bit plane j to the lazy bit plane L: $D2L = j - L$ in Equation (3.10). Figure 3-1 gives an example of

the approximate bit probabilities of the non lazy bit planes (from the 5^{th} bit plane to the 3^{rd} bit plane) for a Laplacian distributed source $\theta = e^{-12}$ and $L = 3$ whose *pdf* is given in Figure 3-1 (a). In each figure (b)-(d), the sum of the area of the shaded region represents the probability of bit equals to one in their corresponding bit plane.



(a) Laplacian *pdf* ($\theta = e^{-12}$, L=3)

(b) Shaded area: $Q_5 = 0.0588$ (5^{th} bp)

(c) Shaded area: $Q_4 = 0.2$ (4^{th} bp)

(d) Shaded area: $Q_3 = 0.3333$ (3^{th} bp)

Figure 3-1 Bit plane approximate probability Q_j example

The L parameter is also easy and practical to obtain by solving Equation (3.11). The bits are then input into the arithmetic coder and encoded with their corresponding approximate probabilities. In addition, only those bits in the *non lazy bit planes* are encoded by the static arithmetic coder because those bits in the

lazy bit planes have a probability of 1/2, and they can be output directly without compression.

It is said in [25] that the BPGC actually can give an identical expected length to that of Golomb code with parameter 2^L for non-negative geometrically distributed integer when the parameter L is greater or equal than 0. But for those very low entropy content sources which may result in $L < 0$, it may not perform well. In addition, by exploiting the statistical properties of the input Laplacian distributed sources, the BPGC algorithm achieves a rate distortion performance which is essentially comparable to an optimal non-scalable scalar quantizer.

BPGC also has a complexity level which is suitable for practical implementation. The calculation of parameter L can be implemented using the C program described in [5]. The static arithmetic coder is also simple and easy to implement as discussed in [24][32].

3.1.2. BPGC used in AAZ

Advanced Audio Zip (AAZ), a lossy to lossless scalable audio coding technology, which is recently been adopted as the reference model for the MPEG-4 Audio Scalable to Lossless Coding, is presented in [26].

AAZ provides backward compatibility by embedding an MPEG AAC (Advanced Audio Coding) bitstream, a widely adopted lossy audio coder. It also provides the functionalities that people previously have to resort to several audio compression technologies like lossy audio coding, lossless audio coding and

scalable audio coding, all in a single framework but without any compromise in terms of coding efficiency or implementation complexity.

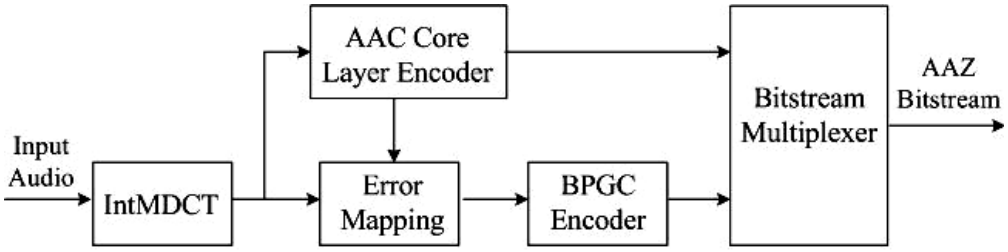


Figure 3-2 Structure of AAZ encoder

The structure of the AAZ encoder is illustrated in Figure 3-2. As indicated in the figure, AAZ consists of two distinguishable layers, one is the perceptual core layer, which has an MPEG-4 AAC audio coder to generate the lossy portion of the embedded bitstream, and the other is the Lossless Enhancement layer (LLE) where the lossy to lossless bitstream is produced. The BPGC encoder is used here to operate on the Laplacian distributed residual IntMDCT (reversible integer MDCT) coefficients which are obtained by an error mapping procedure after the perceptually encoding of the AAC coder. The bitstream generated by the AAC coder represents the minimum rate for the final lossy or lossless bitstream while the BPGC encodes the residual coefficients bit plane by bit plane as described in last section to generate the embedded enhancement bitstream and these two bitstreams are finally fed to the bitstream multiplexer to be the final lossy to lossless bitstream [26].

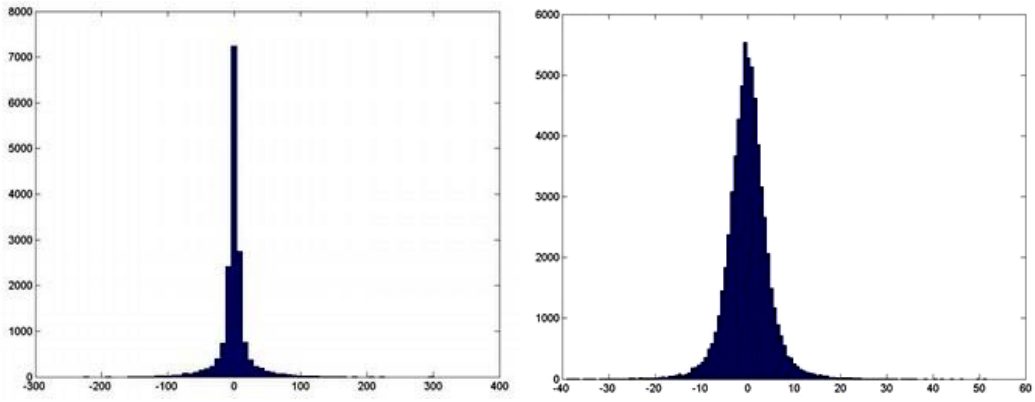
By implementing the lossy coder AAC and the BPGC algorithm which acts as an audio embedded bitstream enhancement scheme, AAZ provides fine grain bitrate scalability without affecting compression performance and also

maintaining a reasonable computational complexity.

3.1.3. Using BPGC in scalable image coding

As discussed before, BPGC is suitable to generate the embedded bitstream for i.i.d. Laplacian distributed sources. The residual IntMDCT coefficients in AAZ have Laplacian distribution. In addition, because the audio signal is a one-dimensional signal where the neighborhood coefficients have low correlations, it can be regarded as near i.i.d. Laplacian distributed.

As reported in many research articles, the wavelet transformed coefficients in high frequency bands, i.e., subbands HL, LH and HH tend to follow the Laplacian distribution [28][29]. The histograms of 3-level decomposition 9/7 Daubechies wavelet coefficients of image *lena* in subband HL_2 and HH_3 are plotted in Figure 3-3. From the figure, we can see that large portion of the wavelet coefficients are around the value zero, and the number of larger magnitude coefficients is exponentially decreasing, which are the characteristic of Laplacian distributed symbols.



(a) (b)

Figure 3-3 Histogram of wavelet coefficients in (a) HL_2 subband; (b) LH_3 subband

Based on the audio scalable coder AAZ, we investigate the possibility of applying BPGC to scalable image coding because these signals to be encoded have something in common, that is, they all tend to follow a Laplacian distribution.

In scalable audio coding, the good performance of BPGC is based on the constraint that the coding source is nearly i.i.d.. However, the BPGC algorithm is not directly useful for coding wavelet coefficients. The spatial dependencies of image wavelet coefficients are quite heavy and that is why many image coders like EZW, SPIHT and EBCOT adopt an adaptive arithmetic coding procedure. Obviously, the BPGC static probability model whose probability is specified by only *D2L* would obviously lose some coding efficiency. Bits in the wavelet coefficients bit planes are significantly affected by nearby coefficients. For example, it is more likely for the current bit to be '1' when most bits of the corresponding neighbor coefficients bit planes are '1'.

However, fortunately, the BPGC algorithm can be easily combined with context modeling techniques to explore the spatial correlation of the input signal. In fact, the extension work on MPEG-4 Audio Scalable Lossless Coding [27] also includes the idea of combining the context technique with the BPGC algorithm, which is called the Context-based Bit Plane Arithmetic Code (CB-BPAC) method. CB-BPAC improves coding efficiency by exploiting the dependencies of the probability distribution of the bit plane symbols of residual IntMDCT coefficients to their frequency locations, the significance states of their adjacent spectral

samples, and their relationship to the lazy plane parameter *L*. The reported overall improvement in lossless audio coding performance is 0.83%.

Thus, if the image context modeling techniques are combined with the BPGC algorithm, it may bring a scalable image coder with efficient compression and also low complexity.

3.2. Context modeling

As mentioned before, the BPGC algorithm specifies a static probability model for the bits in coefficient bit planes, where the bit probability varies according to the distance of the bit plane to the lazy bit plane (*D2L*).

In order to make the static probability model fit the source samples better, spatial correlation is explored by considering the coefficient neighborhood significance contexts. So, firstly, the probabilities of the bits in different bit planes differ from each other by having different *D2L*. And secondly, for a given bit plane, bits probabilities are no longer the same and they are determined by their neighborhood coefficients' significance contexts.

3.2.1. Distance to lazy bit plane

There are 7 *D2L* contexts used in the proposed scalable image coder. The detailed descriptions of these contexts are listed in Table 3-1. *D2L* context 0 represents the lazy bit planes that have the *D2L* value equal or less than -3, and the probability assigned for bits belonging to this context is 1/2. Contexts 1 to 6 are for the non lazy bit planes and these bit planes are assigned with skew bit

probabilities. Note that in Table 3-1 we can see that *D2L* context 6 is for those bit planes with *D2L* greater or equal than 3. Those bit planes which have larger *D2L* are not assigned new contexts. This is because for a Laplacian distributed source, most of the m^{th} bit planes (the most significant bit plane) have *D2L* less than 4. Additionally, for those bit planes with *D2L* greater than 3, the bit probability is so small that if implemented by integer arithmetic coding they can be approximated as the same integer. Grouping them together can reduce the complexity of the context code book.

Table 3-1 *D2L* contexts

Context No.	0	1	2	3	4	5	6
D2L	≤-3	-2	-1	0	1	2	≥3

Note that there is a little difference from the BPGC static probability model we discussed before. In the previous discussion, we considered all the bit planes which have a *D2L* less than 0 as lazy bit planes. However, it is found in our experiments that for image wavelet coefficients, the bits in some of the bit planes which are near the lazy bit plane *L* but have a lower order do not appear to have a uniform distribution. In order to model the wavelet coefficients more accurately, they should be assigned with skew probabilities similar to those bit planes which are of higher order than the lazy bit plane *L*, but obviously not as skewed as those bit planes.

An example of the bit plane coding relating to the *D2L* concept is given in Table 3-2.

Table 3-2 *D2L* context bit plane coding examples

Context No.	6	5	4	3	2	1	0
D2L	3	2	1	0	-1	-1	-3, -4, -5, …
m:8 L:6		8	7	6	5	4	3, 2, 1, 0
m:9 L:6	9	8	7	6	5	4	3, 2, 1, 0
m:7 L:4	7	6	5	4	3	2	1, 0

Basically, the more the number of the *D2L* contexts, the more accurate the model is. However, larger number of contexts leads to larger probability codebooks (lists of bit probabilities), and increasing the probability codebooks will not result in significant improvement in compression performance.

3.2.2. Neighborhood significant states

Another context which should be taken into consideration is the significant states of the neighborhood coefficients. Many image compression systems design delicate sample neighborhood contexts. Part of the neighborhood contexts defined in EBCOT [7] is included in the proposed scalable image coder.

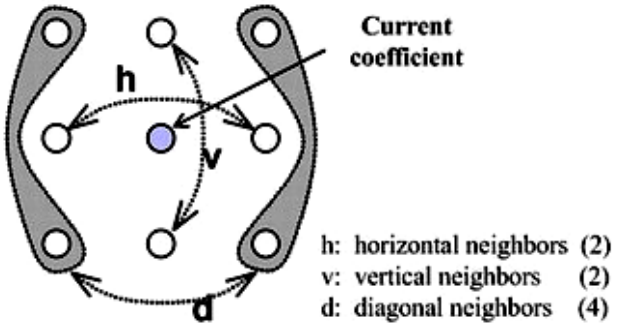


Figure 3-4 Eight neighbors for the current wavelet coefficient

The eight adjacent neighbors of the current coefficient are illustrated in Figure 3-4. Two of them are horizontal neighbors ***h***; the two vertical neighbors are ***v***; and

d indicates the four diagonal neighbors.

When encoding, each coefficient has an associated binary state variable named significance state. Significance state variables are initialized to an insignificant state of 0, and may change to a significant state of 1 during the process of bit plane coding. The significance state context for a given current coefficient is the binary vector consisting of the significance states of its eight neighbor coefficients. Any neighborhood coefficient which lies outside of the independent code block is considered as insignificant.

In general, the given coefficient may have $2^8 = 256$ possible context vectors. However, it is not practical to have so many context vectors. These are clustered into a small number of contexts according to the rules specified in Table 3-3 and Table 3-4. Similar to bit plane coding in EZW, SPIHT and EBCOT algorithm, the proposed coder includes a significance coding pass and a magnitude refinement coding pass for a certain bit plane and each of them is specified a list of contexts.

The contexts included in the significant coding pass are listed in Table 3-3. As shown in the table, there are 9 contexts defined based on how many and which neighbor coefficients are significant. In addition, the mapping of the neighborhood significance states to the contexts also depends on which subband these coefficients are in, i.e., because different subbands have different edge properties. The HL subband tends to be vertical oriented edge; the edge in the LH subband is mostly horizontal; and the HH subband consists of diagonal edges.

Table 3-3 Contexts for the significant coding pass (if a coefficient is significant, it is given a 1 value for the creation of the context, otherwise a 0 value; - means *do not care*)

LH subband (also used for LL subband) (vertically high-pass)				HL subband (horizontally high-pass)				HH subband (diagonally high-pass)		
context	∑h	∑v	∑d	context	∑h	∑v	∑d	context	∑(h+v)	∑d
8	2	-	-	8	-	2	-	8	-	≥3
7	1	≥1	-	7	≥1	1	-	7	≥1	2
6	1	0	≥1	6	0	1	≥1	6	0	2
5	1	0	0	5	0	1	0	5	≥2	1
4	0	2	-	4	2	0	-	4	1	1
3	0	1	-	3	1	0	-	3	0	1
2	0	0	≥2	2	0	0	≥2	2	≥2	0
1	0	0	1	1	0	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0

Table 3-4 shows the three contexts used in the magnitude refinement coding pass. The magnitude refinement coding pass encodes the bits from coefficients which are already significant in the previous bit planes. The contexts used are determined by the summation of the significance states of the eight neighbors and also depend on whether this coefficient is magnitude refined in the previous bit planes or not. Unlike the contexts used in the significant coding pass which are differently defined in subbands because of edge orientation, the refinement contexts are similar among all the subbands and thus are clustered together to only 3 contexts.

Table 3-4 Contexts for the magnitude refinement pass

Context	∑h + ∑v + ∑d	First refinement for this coefficient
0	-	false
1	≥1	true
2	0	true

In EBCOT algorithm, there are also 5 contexts designed for sign coding, which is determined by the significance states and the positive or negative sign symbols of the four horizontal and vertical neighbors [6][7]. However, in the BPGC model, sign bits are simplified as uniformly distributed in order to reduce computational complexity. No contexts are specified for sign coding, and they are output directly to the coded bitstream without any compression.

As described above, we modify the simple BPGC probability model in Section 3.1.1 by combining with the neighborhood contexts. Except for the 7 *D2L* contexts, we add 9 contexts for the significant coding pass and 3 contexts for the magnitude refinement coding pass. Codebooks which contain these probabilities related to the *D2L* and neighborhood contexts are trained offline from large sets of code blocks of natural image wavelet coefficients. They are then pre-saved in both the encoder and the decoder as the prior knowledge for compression.

3.3. Context-based Bit Plane Golomb Coding

By incorporating the image context modeling techniques, we extend the BPGC algorithm to the proposed coder, Context-based BPGC (CB-BPGC). Similar to EBCOT, CB-BPGC acts as an embedded image entropy coder that is applied after the wavelet transform and quantization like in the standard JPEG2000. CB-BPGC differs from EBCOT mainly in the entropy coding *Tier* 1 block coding and uses the same Post Compression Rate Distortion Optimization algorithm (PCRD) as in EBCOT *Tier* 2 to organize the bitstream after embedded block coding.

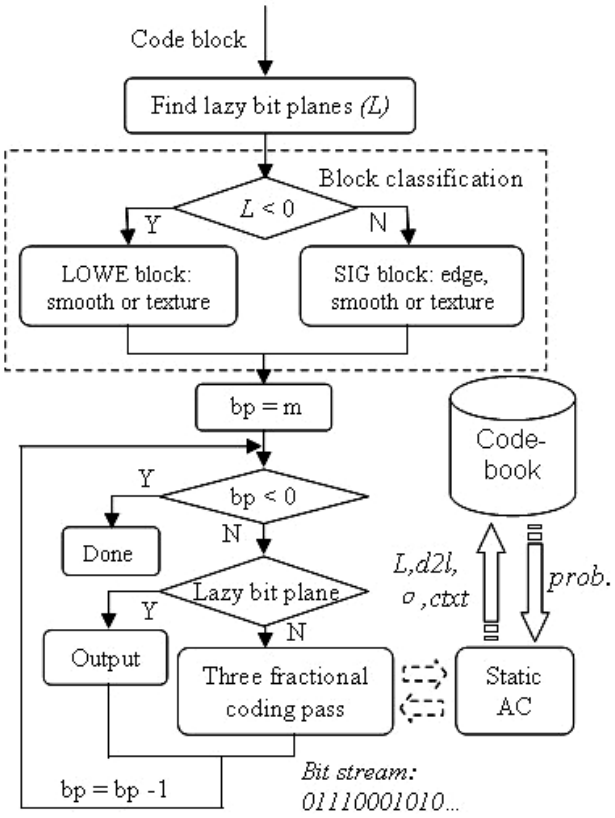


Figure 3-5 Context based BPGC encoding a code block

The process of embedded block coding of a code block coefficients is illustrated in Figure 3-5. As shown in the figure, the first step is to calculate the lazy bit plane parameter *L*. After finding *L*, a procedure of block classification is applied in order to model the individual code blocks in a better way, which we will discuss later. After classification, CB-BPGC starts bit plane coding from the most significant bit plane *m*. The scanning order is the same as described in EBCOT, stripe by stripe in a raster order.

According to their *D2L* contexts, for non lazy bit planes whose *D2L* context numbers are not 0, CB-BPGC applies the three fractional bit plane coding passes, significant coding pass (SIG), magnitude refinement coding pass (MAR) and then

clear up coding pass (CLU), bit plane by bit plane, to get a finer embedded bitstream. The static binary arithmetic coder then compresses all these bits with the look-up probabilities from the codebooks.

After encoding all the non lazy bit planes, CB-BPGC simply outputs the raw bits in the lazy bit planes. In order to guarantee fine grain scalability, the coder first outputs those bits with corresponding coefficients which do not become significant in the previous bit plane (significant coding), and secondly adds the bits from the coefficients which are already significant (refinement coding).

The decoder then simply mimics the encoding process and decodes the bits in bit planes according to the compressed bitstream and the corresponding look-up probability. It should be pointed out that in CB-BPGC the encoding process encodes code blocks with the lazy bit plane parameter L and the block classification parameter σ. These parameters are not directly accessible to the decoder. Thus, they have to be transmitted to the decoder as side information. In our implementation, these parameters are included in the packet header bitstream when the current code block is included in the bitstream the first time.

We will now start to discuss the block classification process. Observations in our experiments indicate that bit probability codebook is slightly different for code blocks with different features and to assign different probability codebooks to those classes is very important.

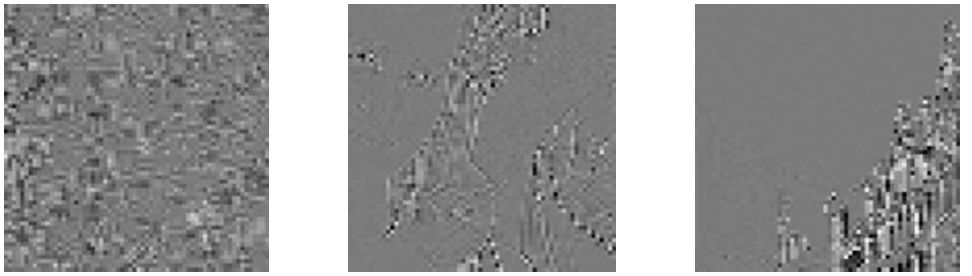
For example, code blocks with quite small lazy bit plane parameter L, especially those blocks with $L < 0$, are mainly blocks in the higher frequency and

higher resolution subbands. They often consist of mainly values around zero, but a few of these coefficients have very large magnitude. These blocks have very low entropy and are called *LOWE* blocks. The *D2L* and neighborhood significance state contexts related bit probabilities of the *LOWE* blocks are quite different compared to those blocks with $L \geq 0$, called *SIG* blocks, which contain many coefficients with large magnitude and these coefficients have much more exponential-like distributions. Different codebooks should be applied to these different block types.

Furthermore, both the *SIG* and *LOWE* blocks can be classified into more specific kinds of block types in order to model the coefficients better. For the *SIG* blocks, three classes appear with slightly different probability codebooks. An example of these classes is illustrated in the first row of Figure 3-6.

The three 64×64 blocks in the figure have the same most significant bit plane $m = 6$ and the same lazy bit plane parameter $L = 3$, which means that for a given bit plane they have the same *D2L* value. But the left block is smooth, and the coefficients with large magnitude spread over the whole block. The middle one seems more textural, where the coefficients with large magnitude appear in small irregular clusters and so do the coefficients with small magnitude. The right block then contains obvious edges, where edges divide the block into smooth regions with large magnitude coefficients or small magnitude coefficients. Because of these distinct local properties, these block coefficients have different neighborhood significance contexts related bit probabilities, i.e. these three classes

should have different codebooks.



5	5	5	5	4	6	5	5
6	5	5	5	5	5	5	5
6	6	5	5	5	6	5	5
6	5	6	5	5	5	5	5
6	6	5	6	5	5	5	5
6	5	6	6	5	5	6	5
5	5	5	5	5	6	6	5
5	6	6	5	6	5	5	5

(a)

3	2	2	5	6	6	3	2
6	3	4	5	5	4	3	4
4	5	6	6	6	3	3	4
4	2	5	5	3	4	5	3
3	5	5	5	3	6	5	6
2	6	5	4	5	6	6	4
3	6	5	5	5	5	6	6
2	3	4	6	6	6	5	6

(b)

1	1	1	1	1	1	1	6
1	1	1	1	1	1	1	6
1	1	1	1	1	1	6	6
2	2	1	1	1	2	6	6
2	2	2	2	1	6	6	6
2	2	4	3	6	6	6	6
2	2	5	6	6	6	6	6
4	3	6	6	6	6	6	6

(c)

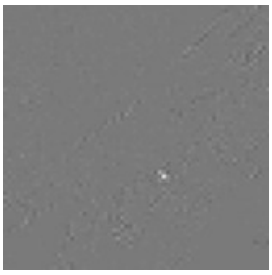
Figure 3-6 Example of three types of *SIG* code blocks with size 64×64 (the first row, coefficients range [-127, 127], white color represents positive large magnitude data and black color indicates negative large magnitude.) and their corresponding *subm* matrixes (8×8) (the second row): (a) smooth block, σ= 0.4869; (b) texture-like block, σ= 1.3330; (c) block with edge, σ= 2.2537.

If each block in Figure 3-6 is divided into smaller 8×8 sub-blocks, and we calculate the most significant bit plane $subm_{x,y}$ of each sub-block (*x, y* indicate the sub-block horizontal and vertical indices respectively) as shown in the second row of Figure 3-6, we can see that the smooth block has a smaller σ, the textual block has a median σ and the edge block has a larger σ, where σ is the standard deviation of the *subm* array given by,

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{x,y}\left(subm_{x,y} - \overline{subm}\right)^2} \tag{3.12}$$

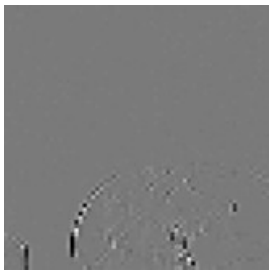
$$\overline{subm} = \frac{1}{n \times m}\sum_{x}^{n}\sum_{y}^{m} subm_{x,y} \tag{3.13}$$

Similarly, *LOWE* blocks also can be divided into two classes (smooth one and texture-like one) according to different σ values as shown in Figure 3-7.



0	1	1	1	0	0	2	2
0	1	1	1	1	2	2	3
2	1	1	1	2	2	2	2
1	1	1	2	0	2	2	1
2	1	2	1	2	2	2	2
2	2	2	2	5	2	2	3
3	2	2	2	2	2	1	0
0	2	2	3	1	1	0	2

(a)



0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	2	1	1	1
0	0	4	3	4	4	2	2
2	1	5	3	3	4	5	2
5	3	5	4	3	5	2	2

(b))

Figure 3-7 Example of two types of *LOWE* code blocks with size 64×64 (the first row, coefficients range [-63, 63], white color represents positive large magnitude data and black color indicates negative large magnitude.) and their corresponding *subm* matrixes (8×8) (the second row): (a) smooth block, σ = 0.9063; (b) texture-like block, σ = 1.7090

The thresholds of the parameter σ for both *SIG* and *LOWE* blocks are then trained to classify all the blocks. Blocks with the same class share the same codebook discussed in the last subsection. We can see that the classification of the blocks according to the parameter σ is very coarse. Obviously, some misclassification may happen. However, the classification measurement based on parameter σ is very simple and is practical to be implemented in the image compression systems.

3.4. Experimental results

The proposed coder CB-BPGC is implemented with the well known Java

implementation of the JPEG2000 standard *JJ2000* [11], where a different block coding is used in CB-BPGC compared to JPEG2000. The codebooks are trained from a large number of natural image wavelet coefficient code blocks with size 64×64. A typical set of grayscale test images of JPEG2000, such as *lena, fruits, cafe,* etc, are used to evaluate the coding performance of the CB-BPGC coder compared to JPEG2000.

3.4.1. Lossless coding

Table 3-5 shows the lossless compression performance at different code block sizes: 64×64, 32×32 and 16×16 for both the JPEG2000 standard and the proposed CB-BPGC coder. The images are encoded together with the 5 level wavelet decomposition of the reversible 5/3 LeGall filter. A similar coding performance comparison for the images compressed by the irreversible 9/7 Daubechies filter is illustrated in Table 3-6.

The numbers of bits per pixel for each losslessly compressed image by JPEG2000 and CB-BPGC are listed in both tables. The positive numbers in the *percentage* column indicate the percentage of CB-BPGC better than JPEG2000 and the negative ones are the reverse.

The average compression results show that CB-BPGC is more efficient than JPEG2000. For the reversible wavelet 5/3 filter, CB-BPGC outperforms JPEG2000 by 0.75% for code block size 64×64, 1.40% for code block size 32×32 and 2.56% for block size 16×16 on average. For the irreversible wavelet 9/7 filter,

CB-BPGC is better than JPEG2000 by 1.06% for code block size 64×64, 1.69% for code block size 32×32 and 2.83% for code block size 16×16 on average.

Note from the tables that the compression performance is especially improved for those images which seem harder to compress, e.g. *baboon* and *cafe*. For those complicated texture-like blocks, e.g., the fine hair in *baboon* and the chaotic buildings and tables in *cafe*, the weaker performance of the adaptive coder arises from its inability to fully exploit context behaviors because of the likelihood of greater variability between block coefficients. As such, the CB-BPGC, which simply provides static bit probabilities according to the *D2L*, the neighbor significance context and the block type parameters, has an edge over the adaptive coder as the former is insensitive to such context variations.

In addition, JPEG2000 loses more efficiency in the case of smaller code block size, e.g. when code block size is 64×64, CB-BPGC is 0.75% better on average for the 5/3 filter but for block size with 16×16, CB-BPGC is 2.56% better. When a smaller block size is used, the number of the coefficients to be encoded is less. The adaptive coder has to restart context adaptation procedure for each code block because of independent block coding; therefore, the number of coefficients in the smaller code block may not be sufficient for the adaptive coder to adapt to the block properties well before the end of encoding process.

Table 3-5 Comparison of the lossless compression performance for 5 level wavelet decomposition of the reversible 5/3 LeGall DWT between JPEG2000 and CB-BPGC (bit per pixel)

Images	Resolution	64×64			32×32			16×16		
		J2K	CB-BPGC	Percentage	J2K	CB-BPGC	Percentage	J2K	CB-BPGC	Percentage
baboon	500×480	6.166	6.020	2.36%	6.277	6.106	2.72%	6.626	6.412	3.22%
barb	720×576	6.249	6.143	1.69%	6.367	6.231	2.13%	6.728	6.553	2.61%
fruits	640×512	4.149	4.168	-0.46%	4.245	4.229	0.38%	4.538	4.451	1.91%
goldhill	720×576	4.645	4.609	0.78%	4.741	4.674	1.42%	5.058	4.937	2.39%
lena	512×512	4.620	4.568	1.12%	4.714	4.629	1.82%	5.022	4.871	3.01%
monarch	768×512	3.845	3.894	-1.28%	3.944	3.940	0.08%	4.237	4.150	2.04%
woman	512×640	4.238	4.234	0.10%	4.329	4.306	0.54%	4.619	4.532	1.89%
café	1024×1280	5.673	5.570	1.80%	5.791	5.671	2.07%	6.148	5.966	2.95%
tool	1280×1024	4.402	4.414	-0.28%	4.509	4.473	0.79%	4.826	4.708	2.44%
actors	1280×1024	5.408	5.320	1.62%	5.522	5.409	2.05%	5.873	5.690	3.12%
average		4.940	4.894	0.75%	5.044	4.967	1.40%	5.368	5.227	2.56%

Table 3-6 Comparison of the lossless compression performance for 5 level wavelet decomposition of the irreversible 9/7 Daubechies DWT between JPEG2000 and CB-BPGC (bit per pixel)

Images	Resolution	64×64			32×32			16×16		
		J2K	CB-BPGC	Percentage	J2K	CB-BPGC	Percentage	J2K	CB-BPGC	Percentage
baboon	500×480	4.924	4.819	2.13%	5.024	4.895	2.56%	5.342	5.179	3.05%
barb	720×576	4.993	4.893	2.00%	5.098	4.979	2.33%	5.421	5.267	2.84%
fruits	640×512	2.606	2.609	-0.11%	2.676	2.672	0.13%	2.893	2.844	1.70%
goldhill	720×576	3.258	3.242	0.49%	3.345	3.304	1.21%	3.616	3.531	2.37%
lena	512×512	3.171	3.124	1.47%	3.253	3.180	2.24%	3.516	3.395	3.43%
monarch	768×512	2.134	2.145	-0.50%	2.503	2.189	0.85%	2.756	2.356	2.83%
woman	512×640	2.654	2.641	0.48%	2.723	2.692	1.12%	2.943	2.888	1.88%
café	1024×1280	4.313	4.244	1.60%	4.418	4.336	1.87%	4.735	4.605	2.76%
tool	1280×1024	2.829	2.800	1.02%	2.917	2.856	2.11%	3.172	3.052	3.78%
actors	1280×1024	4.071	3.988	2.04%	4.172	4.067	2.53%	4.481	4.316	3.69%
average		3.495	3.451	1.06%	3.613	3.517	1.69%	3.888	3.743	2.83%

The CB-BPGC coder which uses static coding does not have such problems. In fact, when the code block is smaller, the CB-BPGC appears to model the coefficients better. The estimation of the lazy bit plane parameter *L* will be more accurate. For example, for blocks with edges in Figure 3-6, if the block size is specified as 16×16, i.e., there are 8 code blocks in this 64×64 image, some of these 16×16 blocks will be classified as smooth. These smooth 16×16 blocks with many larger magnitude coefficients will have larger lazy bit plane parameter *L* compared to those with smaller *L* and also smaller *m* (the most significant bit plane parameter). This possibly makes the compression more efficient than that for block size 64×64.

Table 3-7 Image *Cafe* (512×640) block coding performance, resolution level 0~4, 31 code blocks (5 level wavelet reversible decomposition, block size 64×64)

	Numbers of Bit planes	Byte saved
All bit planes	230	1655
Non-lazy bit planes	159	1235
Lazy bit planes	71	420

Table 3-7 gives an example for the bit plane compression results of the coefficients in image *cafe* levels 0~4 decomposed subbands. For the non-lazy bit planes, CB-BPGC removes more redundancy by using the proposed model and therefore saves bytes in compression. The lazy bit planes performance in the table also shows that it is more efficient to output the raw bits in these bit planes instead of adaptive coding. This is corroborated by the coding performance comparison of JPEG2000, JPEG2000 with lazy coding, and CB-BPGC given in Table 3-8.

As shown in Table 3-8, for most of the test images, JPEG2000 performs better when its lazy coding mode is invoked, and CB-BPGC performs better than the JPEG2000 with lazy coding. This is because although JPEG2000 has lazy coding

mode to directly output these raw bits in the lower order bit planes (fixed 4 bit planes after the most significant bit plane), it has no systematic way to tell from which bit plane a lazy coding is more efficient. However, in CB-BPGC, the lazy bit plane parameter *L* gives an approximate measurement for that.

Table 3-8 Comparison of lossless coding performance (reversible 5 level decomposition, block size 64×64) of JPEG2000, JPEG2000 with lazy coding and CB-BPGC

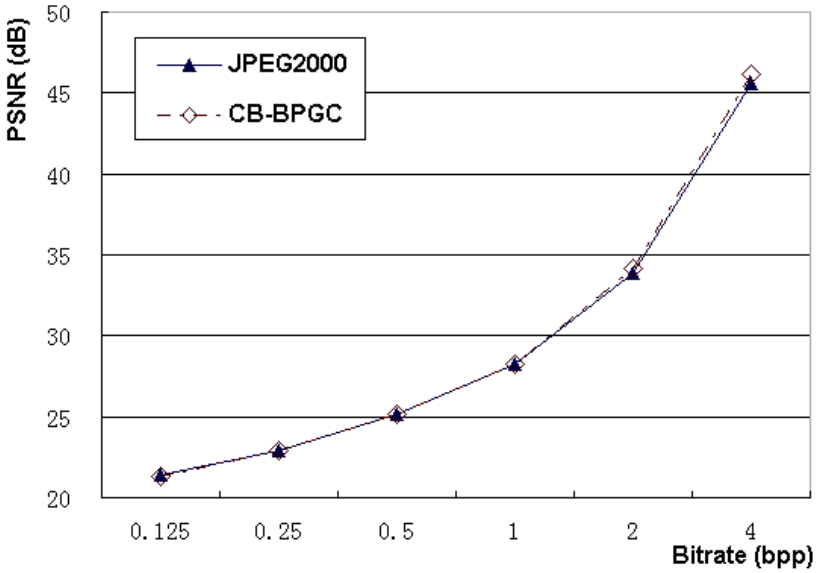
Images	Size	J2K	J2K with lazy coding	CB-BPGC
baboon	500×480	184966	182601	**180604**
barb	720×576	323931	319238	**318441**
fruits	640×512	**169941**	169946	170734
goldhill	720×576	240772	239028	**238908**
lena	512×512	151389	150561	**149689**
café	1024×1280	929388	926601	**912618**
actors	1280×1024	886005	882280	**871694**

3.4.2. Lossy coding

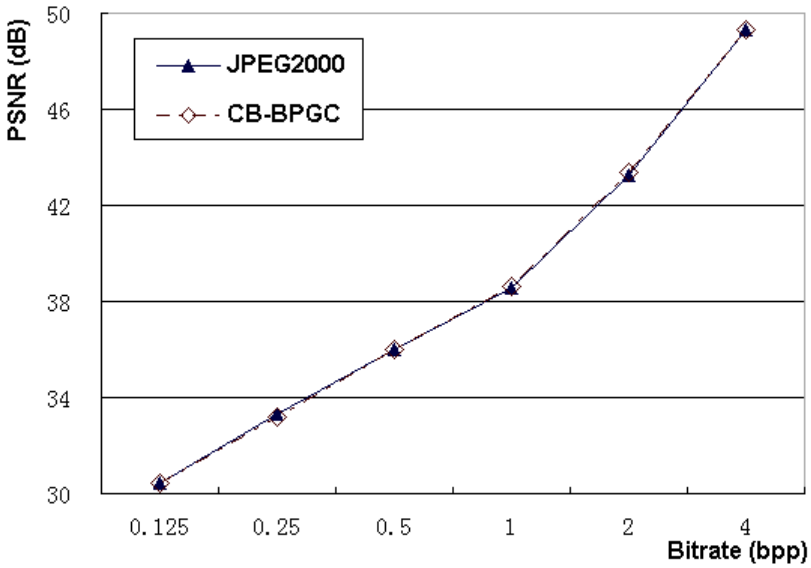
The lossy compression performances of the test images *lena, baboon,* and *actors* at code block sizes 64×64 and 16×16 are illustrated in Figure 3-8.

They are compressed by the irreversible Daubechies 9/7 filter at 5-level decomposition. The figure shows that the lossy compression performance of CB-BPGC is comparable to that of JPEG2000. The PSNR of CB-BPGC is about 0.1dB for bitrate of 1 bpp and about 0.25dB for bitrate 2bpp on average for code block size 16×16. CB-BPGC outperforms JPEG2000 in terms of PSNR at high bit rates, but at very low bit rates, JPEG2000 is better.

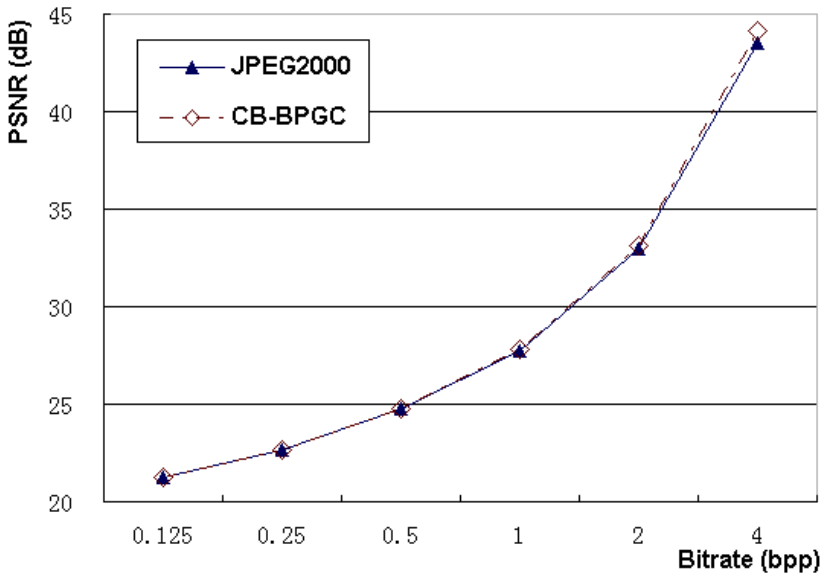
It is probably because these image wavelet transformed coefficients in the LL subband are not near Laplacian distributed. Histogram examples of the coefficients in the LL subband of image *lena* and *peppers* are shown in Figure 3-9.



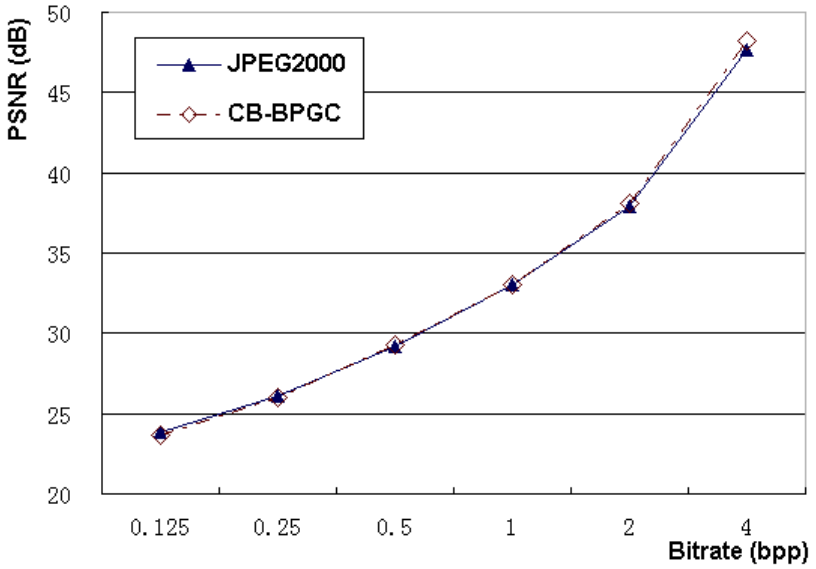
(a) *baboon* (500×480) block size: 64×64



(b) *lena* (512×512) block size: 64×64



(c) *baboon* (500×480) block size: 16×16



(d) *actors* (1280×1024) block size: 16×16

Figure 3-8 Lossy compression performance

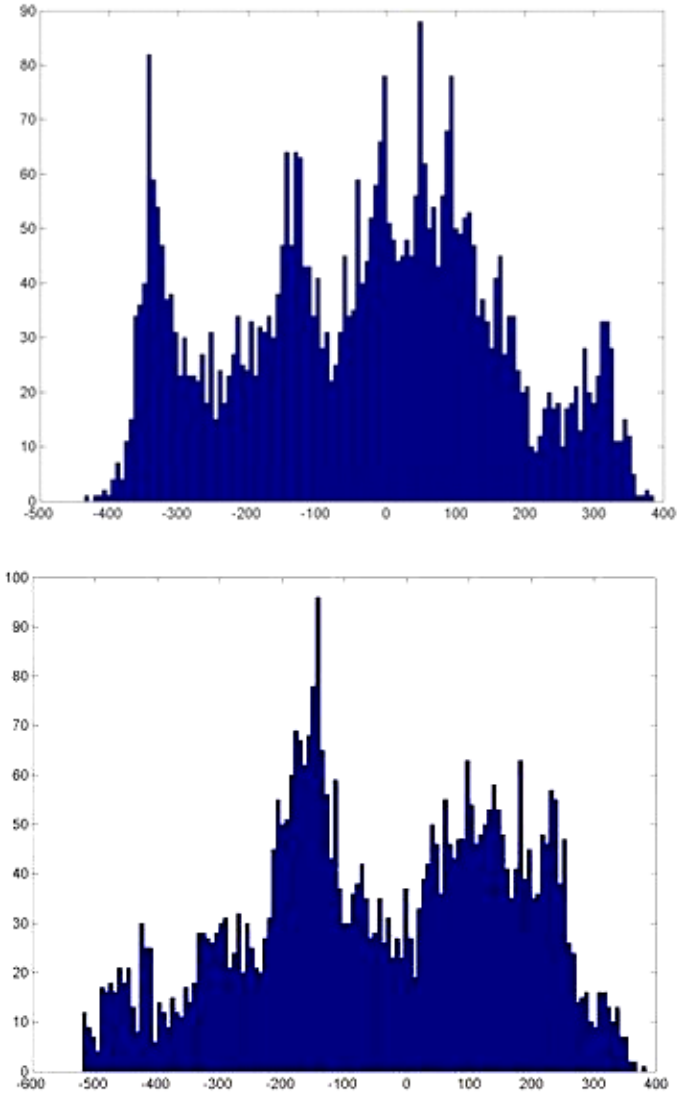


Figure 3-9 Histogram of coefficients in the LL subband of image *lena* 512×512 (top) and image *peppers* 512×512 (down) (Daubechies 9/7 filter, 3 level decomposition)

As can be seen in the figure, the coefficients do not peak around zero value but spread over a large range. The BPGC model which is suitable for Laplacian distribution cannot model them well. However, the inefficient coding of the LL subband coefficients in CB-BPGC significantly affects lossy compression because the LL subband always holds the most important information and they almost always have the priority to be included in the embedded bitstream. Those low bitrate compressed images primarily contains information from the LL subband.

Another possible reason for the modest lossy performance is that the sign bits in

CB-BPGC are uncoded. However, there is likelihood of higher redundancy in the sign bits of the lower frequency subbands, which is left unexploited. Therefore, this also results in compression inefficiency at lower bit rates.

3.4.3. Complexity analysis

We briefly analyze the complexity issues of the proposed CB-BPGC coder. Generally speaking, CB-BPGC has a lower complexity than JPEG2000's entropy coder EBCOT.

As mentioned in Section 3.3, CB-BPGC and EBCOT utilize a similar process of entropy coding by separating it into block coding and post-processing bitstream organization. They differ mainly in the entropy coding *Tier* 1 block coding part, which is the most time consuming part in JPEG2000. The runtime percentages of grey scale image lossless and lossy encoding are reported to be about 71.63% and 52.26% respectively in [30] .

Table 3-9 Average run-time (ms) comparisons for image *lena* and *baboon* (JPEG2000 Java implementation *JJ2000* [11] and Java implementation of CB-BPGC)

	lena (512×512)		*baboon* (500× 480)	
	lossless compression	lossy at 1bpp	lossless compression	lossy at 1bpp
JPEG2000 block coding	537.18	511.03	582.89	559.16
CB-BPGC block coding	452.55	423.36	492.81	460.18
JPEG2000 encoder	752.43	954.33	811.43	1044.87
CB-BPGC encoder	667.25	867.54	723.29	945.95
Runtime pencentage saved for entire encoding	11.32%	9.09%	10.86%	9.47%

Table 3-9 lists the runtime profiles for grey scale images *lena* and *baboon* encoded by both the proposed CB-BPGC coder and JPEG2000. Both lossless

compression and lossy compression at a typical bit rate of 1 bpp are tested. The two encoders apply 5-level wavelet decomposition at code block size 64×64. They are both implemented in Java language (JBuilder). The testing platform is the IBM laptop T42 with 256M RAM 1.6G Hz. Every number in the table is the average results of 200 runs under the same parameters.

Experimental results show that CB-BPGC consumes about 84.58% of the runtime of EBCOT *Tier* 1 block coding for the lossless compression mode and 82.74% for the lossy compression at bitrate of 1 bpp. For the whole encoding process, about 11.6% of the lossless encoding time and 9.28% of the lossy encoding time are saved in CB-BPGC for grey scale images *lena* and *baboon*.

The possible reasons for the reduced runtime of CB-BPGC are as follows. First, directly outputting the sign bits and bits in the lazy bit planes reduces some burden of the context modeling in CB-BPGC. As shown in Table 3-7 *cafe* example, for the levels 0-4 code blocks, 30.9% of the bit planes are directly transmitted lazy bit planes ($D2L \leq 3$). The complexity for these bit planes is obviously reduced relative to adaptive binary arithmetic encoding in JPEG2000. In addition, we can further reduce the complexity for the lazy bit planes by letting more bit planes be lazy bit planes, for example, bit planes with $D2L \leq -1$, where 57.8% of the bit planes in the *cafe* example can be directly output. Experiments also show that the average lossless coding performance is still better than EBCOT by 0.73% and 2.49% for block sizes of 64×64 and 16×16 respectively. This tells us that by sacrificing a little coding efficiency, computation complexity could be much reduced for these bit planes.

Second, the static arithmetic coder is always simpler and of lower computational complexity than the adaptive arithmetic coders by avoiding the probability

adaptive procedure. A recent arithmetic coding complexity scheme [31] shows that the encoding time of a static arithmetic coder is about 58.6% of the MQ coder. Since a static arithmetic coder is used in CB-BPGC while EBCOT uses the adaptive MQ coder, a reduction of computation complexity of CB-PBGC is also achieved.

Additionally, the inclusion of CB-BPGC's special processing steps, the calculation of the lazy bit plane parameter and the code block classification have negligible influence on complexity increase. The lazy bit plane parameter calculation only needs the absolute sum of the block coefficients and the number of the block coefficients. In the process of block classfication, the calculation of the sub-blocks' most significant bit plane numbers can be carried out when the sum A is determined. The classification is then simply by several threshold comparison.

3.5. Discussion

In this chapter, we present the proposed CB-BPGC coder for scalable image compression based on the statistical characteristics of the wavelet coefficients. By combining the embedded bit plane coder BPGC with context modeling techniques, CB-BPGC outperforms JPEG2000 for the lossless compression, and obtains comparable lossy compression performance. In addition, computational complexity of the CB-BPGC is lower than JPEG2000 and the complexity of CB-BPGC can be further reduced by reducing the number of *D2L* contexts.

Generally, lossless compression benefits significantly from efficient lazy bit plane coding. Coefficients from the reversible wavelet filter contains low order bit planes full of uniformly distributed bit symbols, which contain the detailed

information and are reserved in order to guarantee losslessly reconstruction. For images with complicated textures, these bit planes consume a large portion of the codestream, where the bit symbols 0 and 1 have probabilities of 1/2 and they are nearly uncompressible. Compared to the method of adaptive arithmetic coding of these symbols, which blindly adapts to these equal probable symbols, determining the bit plane parameter L according to the block statistical property and then directly transmitting these raw bits in the lazy bit planes will make the encoding process more simple and efficient.

However, for lossy compression, CB-BPGC only provides modest improvement compared to JPEG2000. It is well known that higher order block coefficients bit planes have skew probabilities, i.e. most of the symbols are 0 and they are much spatially correlated. Hence, bits in those bit planes have lower entropy and are more compressible. The adaptive coding methods are then suitable for compressing them. The proposed CB-BPGC probability model for those bit planes carry out compression from another perspective by assigning fixed look-up probabilities from the bit plane probability codebooks. However, because of the inefficient coding of the LL band coefficients, whose codestream often has the priority to be included in the final embedded bitstream, CB-BPGC achieves only comparable lossy compression performance.

As the modeling and ordering of the block coefficients is very important in image compression systems, there are also possible ways to improve compression of the CB-BPGC algorithm. As discussed in Section 3.4.1, CB-BPGC performs much better for those code blocks with smaller block sizes where the lazy bit plane parameter and the assigned bit probabilities more closely match the real situations. Larger sub-blocks require less side information but the static bit

probability model in CB-BPGC may lose some efficiency because of the variation of the local coefficients distribution properties. On the other hand, smaller sub-blocks are more accurately modeled by CB-BPGC, but require more side information to accompany all the smaller sub-blocks. Thus there is a trade-off that would yield a suitable configuration to achieve the best compression performance.

The neighborhood significant state context modeling can also be changed in CB-BPGC. The adjacent coefficients context modeling in the current JPEG2000 and CB-BPGC are the most complicated and time-consuming parts of the encoding process. A novel modeling and ordering of the wavelet coefficients method is proposed in [16]. The neighbor correlations there are modeled by the so-called context template which is very simple and reduces lots of computational complexity. This context modeling technique can be easily implemented in the CB-BPGC framework.

Chapter 4. ERROR RESILIENCE FOR IMAGE TRANSMISSION

Because of the increasing interest in robust image transmission over channels, such as wireless networks and the Internet, error resilience in image communications is becoming more and more important [35]. These unreliable wire or wireless channels may inject errors into the transmitted bitstream. However, a loss or damage of packets in the image delivery may lead to reconstructed images fully or severely damaged. Sometimes even very few errors can cause unpleasant block or ripple effects on the decoded images as shown in Figure 4-1.

In this chapter, we first review some of the error resilient techniques designed at the source coding level, including the error resilience tools used in the standard JPEG2000. Then we present the error resilient techniques of the proposed coder CB-BPGC. A comparison of the error resilient performance between the JPEG2000 and the CB-BPGC is also included in this chapter.

4.1. Error resilience overview

Error resilient techniques have been studied for a long time. Methods such as Automatic Repeat Request (ARQ) allow requests for retransmission of the lost or damaged packages if dialogue between the source and the destination is possible. However, for most real-time applications, such mechanisms often bring unbearable delay or sometimes the dialogue between them is impractical to be set up, such as broadcast applications. Other channel coding methods such as forward

error correction (FEC) can reduce effects of transmission errors but with an increasing complexity, bandwidth or reconstruction delay. Therefore, error resilient techniques at the source coding level are receiving greater attention and are very helpful in improving robust transmission [34].

Figure 4-1 Corrupted images by channel BER 3×10^{-4}(left: encoded by DCT 8×8 block; right: Daubechies 9/7 DWT, block size 64×64)

Error resilient techniques used in image transmission at the source coding level attempt to generate a compressed bitstream which is not vulnerable to channel errors and have the ability of accurate self-detecting and correcting these errors. They include techniques of resynchronization, error resilient entropy coding, e.g. fixed length coding and reversible variable length codes, and some other error correction techniques.

4.1.1. Resynchronization

The most popular and effective error resilient scheme is resynchronization. Almost all the image coding systems featured with error resilience use these techniques. Resynchronization tools attempt to establish the synchronization between the encoder and the decoder when the compressed bitstream is corrupted by transmission errors. They localize the error positions and prevent error propagation. The data between a synchronization point before the error position and the next synchronization point are discarded [35].

One common solution for resynchronization is to insert some unique markers to the encoded bitstreams as boundaries for different layers, different spatial areas, or different bit planes. It is widely used in image compression systems, such as JPEG2000. However, compression is sacrificed a little because these markers used for resynchronization involve additional redundant information.

There are also some schemes which do not need markers in the compressed bitstream but organize the bitstream in another way, such as the error resilient entropy code (EREC) described in [33], which is designed for variable length coded block coding. The idea of the EREC algorithm is to reorganize a group of variable length blocks to constant-length slots. The EREC bitstream is composed of N slots, each of length S_i, and the decoder knows the parameters N and S_i before decoding. EREC greatly reduces error propagation in DCT based image coding because of the use of fixed length slots. For these DCT blocks, the lower frequency coefficients are included in the beginning part of the slot. It is obvious that in order to increase the quality of the reconstructed image, it is preferable that more important information, such as the lower frequency coefficients, can be recovered as early as possible. As such it is advantageous to place the lower frequency coefficients at the beginning of the slot so that when errors occur of latter parts of a slot the crucial information has already been received.

However, the EREC scheme needs a complicated algorithm to reorganize the bitstream into constant length slots and the organized bitstream cannot satisfy the scalability requirement, which is not as convenient as schemes that involve the addition of resynchronization markers.

4.1.2. Variable length coding algorithms resilient to errors

Variable length coding algorithms are widely used in image compression systems because of their efficiency in terms of coding performance compared to fixed length coding strategies. However, unlike the fixed length entropy coder, which encodes every symbol with a fixed length code and enables the decoder to self synchronize when errors occur, the variable length coded bitstream are very sensitive to channel errors and it is always hard to detect the position of errors if no channel error detection methods are used. These errors can then propagate into the following long bitstream sequences and corrupt decoding of the following symbols. Much effort has therefore been spent on designing variable length coding algorithms which has the ability to detect errors and obtain self synchronization.

The reversible variable length coding strategy (RVLC) is one of the approaches. An example of RVLC used for robust image and video transmission is presented in [36], where the codec has the characteristic of being decodable in two directions. The RVLC technique is also included in the MPEG-4 video codec to encode the DCT coefficients of macroblocks corresponding to texture information [37]. Whenever an error is detected in the bitstream, the decoder starts to decode from the end of the bitstream in a reverse direction to continue reconstruction. Hence, robustness is enhanced in the presence of transmission bit error. In addition, these RVLC schemes also involve little or no efficiency loss relative to corresponding non-reversible variable length codes.

There is also a fast synchronization Huffman coding algorithm presented in [38]. The so called suffix-rich Huffman code has a reduced length of error propagation compared to traditional Huffman code. It is thus better self-synchronized and

more resilient to channel errors.

The most popular arithmetic coding algorithm can also be modified with the ability for error detection. The algorithm in [39] presents an approach to introduce a forbidden symbol as an extra alphabet in arithmetic coding and assigned the forbidden symbol with a probability similar to the source alphabets. Apparently, the forbidden symbol is not encoded as an input symbol in the encoding process but is very effective in the decoding procedure. If the bitstream is corrupted, the arithmetic decoding procedure may enter into the interval of the forbidden symbol, i.e. the current decoded symbol is the forbidden symbol, and then an error is detected. The practice of including the extra forbidden symbol will add redundancy to the coding source. But the probability assigned to the forbidden symbol can be adjusted to balance the error detection performance and the compression efficiency it affects. This method is also included in the latest standard JPWL (JPEG2000 for wireless applications) [40] to make the MQ coder more error resilient.

4.1.3. Error correction

Except for the techniques mentioned above which refer to making the compressed bitstream more resilient to errors, there are also some other ways to help improve the quality of the corrupted images, which we call error correction or error concealment techniques. Error correction works as a post-processing procedure after the damaged bitstream is decoded into corrupted images or works together with the decoding procedure when the corrupted images are under reconstruction.

One of the most commonly used methods is the spatial interpolation. In this technique, the lost or damaged coefficients and blocks are interpolated or

predicted from the neighboring correctly decoded symbols [35]. Spatial interpolation can be carried out both in the pixel domain, which means the reconstructed images from the corrupted bitstream, and the frequency domain, which refers to the DCT coefficients or the wavelet subbands coefficients. However, the interpolation will often result in smooth or blur areas and limit the reconstruction of the detailed information in the images, such as edges. In addition, sometimes it will lead to block artifacts, especially for interpolation carried out in the frequency domain, which greatly reduces the image quality.

In order to recover the image details like edges, there are some error concealment schemes based on edge directed filters for wavelet based image compression, such as in [43]. The annoying ripples around the edges in the corrupted images can be removed by the edge directed filter and it results in more pleasant subjective quality images.

In addition to techniques which conduct error correction on coefficients by interpolation or edge filter, schemes based on the prediction of bits in the coefficient bit planes have also been explored. The method in [41] proposes an approach to improve the error resilient ability of JPEG2000. It recovers damaged wavelet coefficient bit plane symbols according to its corresponding cross subbands undamaged coefficient bit planes.

4.2. Error resilience of JPEG2000

As we introduced in Section 2.7, entropy coding in JPEG2000 is achieved by a context based adaptive binary arithmetic bit plane coder (the MQ coder). The MQ coder is a variable length coding method and its encoding process is highly dependant on the state of the coded symbols. A single bit error in the arithmetic

coded bitstream can result in erroneous reconstruction. So, it is very important to maintain synchronization between the encoder and the decoder. To solve this problem, several error resilient tools are provided in JPEG2000.

The error resilience tools adopted in JPEG2000 can be mainly classified into two types, one is error resilient techniques at the packet level and the other is at the entropy coding level [7][34].

In JPEG2000, wavelet subband coefficients are divided into code blocks with certain block size and these code blocks are encoded independently. This data partitioning strategy provides the possibility to prevent propagating errors encountered in a certain code block bitstream to the process of reconstructing other code blocks.

The coded block bitstreams are then organized hierarchically in a structure of packets by subbands, bit planes and blocks according to spatial or quality scalability constraint. Segments from various blocks are collected together in a packet to form the packet body part and preceded by a resynchronization packet header where the header consist of unique markers that never included in the packet bodies. The use of error resilient packet header markers enables the decoder to reestablish block synchronization after bit errors.

JPEG2000 also provides a mechanism where the packet headers can be extracted from every packet and stored in the tile header or the main header, which contain the most important information, such as code block truncation points, bit plane coding parameters and so on. These headers can be transmitted via a more reliable channel in an error free fashion.

The error resilience tools at the entropy coding level of JPEG2000 includes termination coding for each coding pass, reset of contexts, bypass coding and bit

plane coding segment markers at the expense of small losses in compression efficiency. These mechanisms are enabled by mode variation, RESET, CAUSAL, RESTART, SEGMARK, ERTERM and BYPASS. Mode variations are controlled by flags that are included inside headers.

The RESET mode is used to reset the context states, i.e. the probabilities used in the MQ coder, to their initial values at the end of each coding pass. When the RESET is switched off, initialization occurs only prior to the first coding pass of each code block. The reset of the context states may prevent the error propagation by reducing context dependency between coding passes.

The CAUSAL option is defined to allow parallel processing of coding passes and makes the coefficient significant states updating within a single stripe. Thus in this mode, the coefficients within a given stripe are encoded without depending on the values of future stripes.

The RESTART switch makes the MQ coder terminate at the end of each coding pass and restart coding at the next coding pass, which means that every coding pass has separately MQ encoded bitstream segment. The error occurring in the current bit plane may not affect the next coding pass decoding if this mode is specified and the length of each segment is included in the headers. When this mode is switched off, the MQ coder terminates coding only at the end of the current code block.

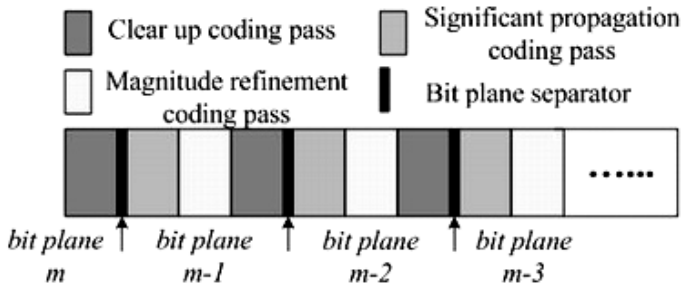


Figure 4-2 JPEG2000 Segment marker for each bit plane

The goal of the SEGMARK mode is to provide segment separators between bit planes. A special four symbol code, “1010”, is inserted at the end of each clear up coding pass to enhance error resilience as illustrated in Figure 4-2. Whenever the special segment separator is wrongly decoded, it indicates that the current bit plane is corrupted by errors and should be discarded.

There is also a very important mode called ERTERM. When the ERTERM is utilized, the encoder adopts a predictable termination policy for each coded segment. Then, the decoder can detect an error has occurred in the arithmetically coded bitsteam segment.

The BYPASS mode is to provide reduced complexity at high bitrates by bypassing coding bits in the significant propagation coding passes and the magnitude refinement coding passes after the first 10 coding passes, i.e. from the $(m\text{-}4)^{th}$ bit plane to the least significant bit plane. These binary symbols are outputted in raw bits.

When the decoder detects errors in a certain bit plane of a code block, JPEG2000 then replaces the current and the following bit planes of the current code block by zeros to prevent error propagation. Apparently, every error resilient tool used here makes the bitstream more resilient to errors by inserting additional information in the bitstream, i.e. at the expense of loss of coding efficiency.

4.3. CB-BPGC error resilience

In this section, we are going to discuss the error resilient tools adopted in the proposed coder CB-BPGC. Most of the tools used here are based on the JPEG2000 error resilient tools at both the packet level and the entropy coding level. However, some modifications are made here.

4.3.1. Synchronization

Similar to JPEG2000, CB-BPGC hierarchically organizes the coded bitstreams by subbands, blocks and bit planes. The same resynchronization markers at the packet level are set up to prevent error propagation.

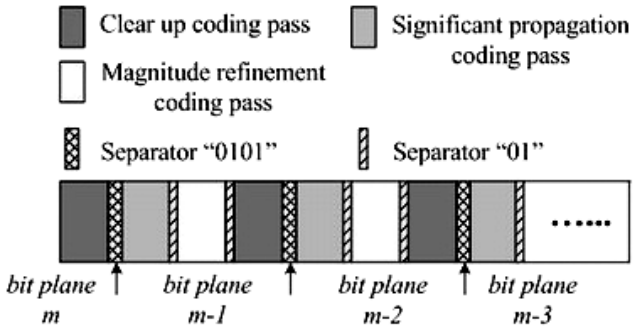


Figure 4-3 CB-BPGC segment markers for bit planes

Figure 4-3 illustrates the error resilience strategies used in CB-BPGC entropy coder for non lazy bit plane coding where three fractional bit plane coding passes are included. The static arithmetic coder terminates at each fractional bit plane coding pass to stop error propagation. The independent coding of the fractional bit planes also enables the so-called bit plane partial decoding which will be discussed in the next section.

A segment marker "0101" is inserted after each clear up coding pass which is also the end of the current bit plane. A segment marker "01" is also added when

each significant propagation coding pass and magnitude refinement coding pass is done. Whenever a mistake appears in decoding these markers, an error is detected.

For the lazy bit planes, because the bits in the bit planes are directly output in two passes, namely, the significant pass and the refinement pass, there is no need to insert a segment marker for every coding pass. Only a segment marker “01” is added after each bit plane. Therefore no extra redundancy for resynchronization is added in CB-BPGC for error resilience.

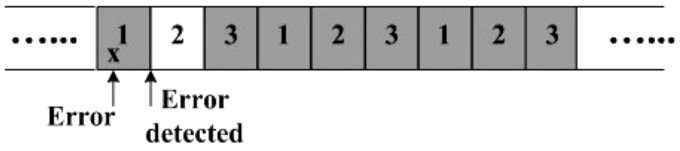
4.3.2. Bit plane partial decoding

Although the error resilient tools specified in JPEG2000 provide a coded bitstream resilient to errors, some improvements can be made. The authors in [44] point out that there are dependencies among the coding passes for a certain code block, where partial decoding of the corrupted bitstream can be added to improve the error resilience performance. For example, if an error is detected in the current magnitude refinement coding pass, instead of setting the current and the remaining bit planes to zeros we can leave the decoded significant propagation coding pass bits.

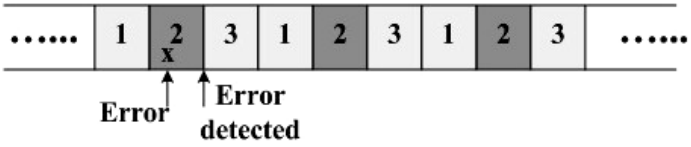
The basic idea of partial decoding is to decode as much as possible of the corrupted bitstream before discarding them. Because CB-BPGC encoded each coding pass in an independent way, it can be conveniently carried out with the idea of partial decoding.

The mechanism used in the CB-BPGC decoder for partial decoding of the bit planes of the error blocks for non-lazy bit planes is illustrated in Figure 4-4. We denote in the figure the significant propagation coding pass as coding pass 1; the magnitude refinement coding pass as coding pass 2; and the clear up coding pass

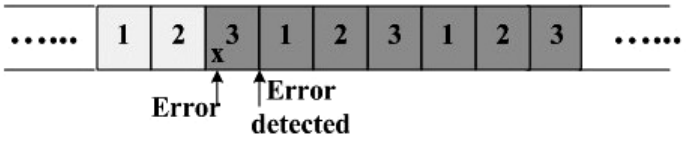
as coding pass 3.



(a) case 1



(b) case 2



(c) case 3

Figure 4-4 CB-BPGC partial decoding for non-lazy bit planes (coding pass 1: significant propagation coding pass; coding pass 2: magnitude refinement coding pass; coding pass 3: clear up coding pass. “x” means error corruption.)

As indicated in the Figure 4-4, there are three cases based on which part of bitstream is corrupted. The partial decoding of each case is as follows,

(a) *Case 1:* error detected in coding pass 1, no further coding passes 1 and 3 can be decoded, but coding pass 2 in the current bit plane can proceed.

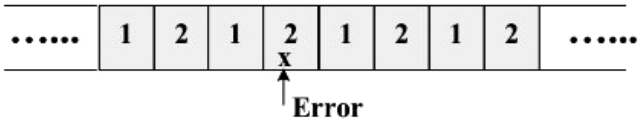
(b) *Case 2:* error detected in coding pass 2, no further coding pass 2 can be decoded, but coding passes 1 and 3 in the current and following bit planes can proceed.

(c) *Case 3:* error detected in coding pass 3, no further coding passes 1, 2 and 3 can be decoded.

The partial decoding of lazy bit planes is illustrated in Figure 4-5. We denote the significant coding pass as coding pass 1 and the refinement coding pass as

coding pass 2.

(a) case 1



(b) case 2

Figure 4-5 CB-BPGC partial decoding for lazy bit planes (coding pass 1: significant propagation coding pass; coding pass 2: magnitude refinement coding pass. "x" means error corruption.)

As shown in the figure, there are two cases which are also classified according to which part of bitstream is attacked.

(a) *Case 1:* error in coding pass 1, coding pass 2 in the current bit plane can proceed, but no further coding passes following cannot be decoded.

(b) *Case 2:* error in coding pass 2, no significant influence on other coding passes, and the coding pass itself can also be reserved because only the bit corrupted by error is wrongly decoded.

Note that the error resilient PSNR gain reported in [44] is based on the assumption that there is an external error detection mechanism to tell the decoder which byte in a certain fractional bit plane is corrupted by errors, which leads to a more complicated partial decoding applied on the fractional bit plane level instead of the bit plane level, i.e. additional information outside of the JPEG2000 decoder helps to guide the decoder to decode much more corrupted coded bitstream. Our test results show that by only using the internal error detection method in CB-BPGC, substantial PSNR improvement can be obtained when the image is

transmitted through a Rayleigh channel.

4.4. Experimental results

The proposed error resilience tools used in CB-BPGC which is described above are evaluated by the set of natural testing images mentioned in Chapter 3. The performance result is compared to the JPEG2000 standard with the entropy coding level error resilient mode RESET, CAUSAL, RESTART, ERTERM, SEGMARK and BYPASS switched on. The compressed image bitsteam transmission is simulated through a wireless Rayleigh fading channel. The results provided in this section are obtained from 500 times realizations over the simulated channel with a given BER for each image in the test set.

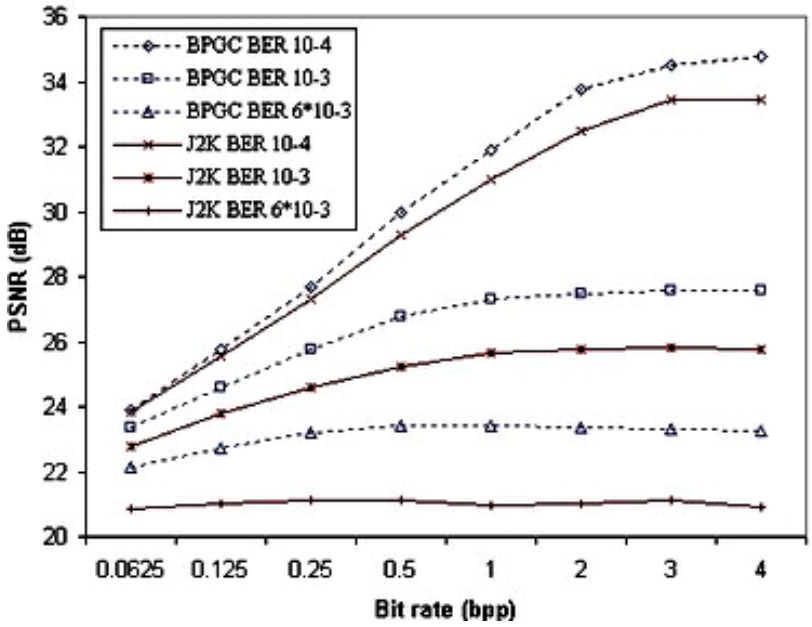


Figure 4-6 Comparison of error resilience performance between JPEG2000 (solid lines) and CB-BPGC (dashed lines) at channel BER 10^{-4}, 10^{-3}, and 6×10^{-3}

Figure 4-6 shows the comparison of average PSNR performance between the CB-BPGC coder and the standard JPEG2000 at channel BER 10^{-4}, 10^{-3} and 6×10^{-3} for different bit rates.

Both the encoders are set with 5-level Daubechies 9/7 wavelet decomposition,

bitstream with resolution-layer-component-position progression order organization, block size: 64×64. For both bitstreams, LL subband layers are protected from error corruption, where the most important information is located in the embedded stream and often assumed to be transmitted through a more reliable channel.

As shown in Figure 4-6, CB-BPGC is more resilient to errors, especially when the channel error bitrate is higher. The improved PSNRs averaged for all the bit rates are 0.731dB, 1.514dB and 2.097dB for BER at 10^{-4}, 10^{-3}, and 6×10^{-3} respectively.

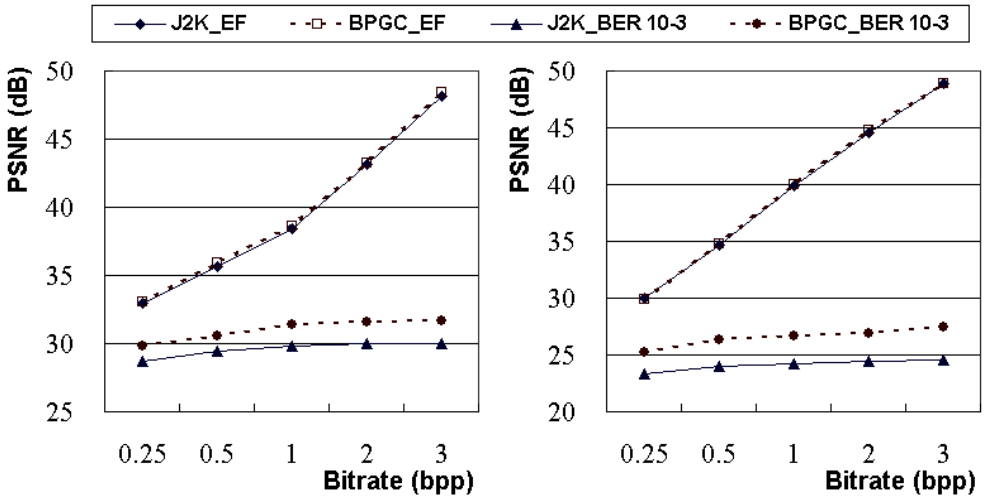


Figure 4-7 PSNR comparison for channel error free and channel BER at 10^{-3} for image *lena* 512×512 (left) and *tools* 1280×1024 (right)

Figure 4-7 gives a further example of the average PSNR comparison of the error free and error corrupted decoding at BER of 10^{-3} for images *lena* and *tools* at several bit rates with code block size 64×64. As shown in the figure, the PSNR improvement of the CB-BPGC is 1.15 dB for image *lena*, 2.25 dB for image *tools* at bit rate 0.5 bpp and can be as much as 1.70 dB for image *lena*, 2.79 dB for image *tools* at bit rate 3 bpp.

Subjective results of some of the images, like *lena*, *bike*, *peppers*, *woman*, etc,

at BER of 10^{-3} and 1 bpp are shown in Figure 4-8. Comparing the two reconstructed images, we can see that CB-BPGC gains not only in better PSNR performance in dBs, but also a substantial improvement of subjective visual effect.

(a) Error free lena (256×256) (b) JPEG2000 (27.002 dB) (c) CB-BPGC (30.957 dB)

(d) Error free bike (256×256) (e) JPEG2000 (22.172 dB) (f) CB-BPGC (25.901 dB)

(g) Error free peppers (256×256) (h) JPEG2000 (27.251 dB) (i) CB-BPGC (29.254 dB)

(j) Error free actors (256×204) (k) JPEG2000 (26.726 dB) (l) CB-BPGC (28.788 dB)

(m) Error free goldhill (256×204) (n) JPEG2000 (27.475 dB) (o) CB-BPGC (30.454 dB)

(p) Error free woman (256×320) (q) JPEG2000 (27.730 dB) (r) CB-BPGC (35.075 dB)

Figure 4-8 Subjective results of image *lena* (a~c), *bike* (d~f), *peppers* (g~i), *actors* (j~l), *goldhill* (m~o) and *woman* (p~r) at bit rate 1 bpp and channel BER 10^{-3}

The improvement in error resilience performance of CB-BPGC is not only gained by adding the partial bit plane decoding which is used to decode the corrupted codestream as much as possible, but also by more efficient compression. As the PCRD algorithm organizes the coded bitstream according to the contribution of reducing distortion, i.e., in a decreasing order, more efficient compression enables CB-BPGC to consume less bytes to embed the coded

bitstream while still providing the equivalent distortion reduction. Hence, when a transmission error occurs, it corrupts the less important bitstream of CB-BPGC and the PSNR result is better. Additionally, directly outputting lazy bit planes also improves error resilience performance. In spite of errors that occur in the lazy bit plane, we can further decode the remaining coefficients because the errors are isolated to certain coefficients instead of propagating to the others.

4.5. Discussion

In this chapter, we present error resilient tools used in the proposed coder CB-BPGC. Compared to the JPEG2000 standard, CB-BPGC is more resilient to transmission errors when simulated over the wireless Rayleigh fading channel. The improved average PSNRs at bit rate 1bpp are 0.918dB, 1.674dB and 2.471dB for channel BER at 10^{-4}, 10^{-3}, and 6×10^{-3} respectively.

The improvement of error resilient performance in CB-BPGC is obtained from efficient scalable coding, bit plane partial decoding and also from direct transmission of the lazy bit planes. Note that in CB-BPGC each bit in the bit plane is entropy coded by a look-up probability from the codebook. When the compressed bitstream is corrupted by the channel errors, the decoder loses synchronization with the encoder. It is then possible for the decoder to reconstruct the corrupted symbols by utilizing the bit probabilities look-up from the codebook to estimate the lost bits. However, the estimation process should be carefully designed in order to avoid artifacts.

Chapter 5. CONCLUSION

Wavelet based image compression schemes are widely used in scalable image coding, In this thesis, we present the proposed wavelet based scalable image entropy coder, namely, Context-based Bit Plane Golomb Coding (CB-BPGC). By utilizing the embedded bit plane coding algorithm, bit plane Golomb coding (BPGC) together with the image context modeling techniques, CB-BPGC explores both the global and local statistical characteristics of the wavelet coefficients blocks

CB-BPGC outperforms JPEG2000 in terms of compression performance. Experimental results show that the proposed coder CB-BPGC achieves a 0.75% better lossless performance for 5-level 5/3 wavelet decomposition at block size 64×64 and 2.56% at block size 16×16. A PSNR improvement of lossy compression performance is also achieved except at very low bit rates.

Besides, because of the partial decoding, the direct transmission of lazy bit planes and the better compression ratio which may lead to corruptions to the less important bitstreams, CB-BPGC is more resilient to transmission errors compared to the JPEG2000 standard. The improved PSNR average performance for all the bit rates is 0.731dB, 1.514dB and 2.097dB for BER at 10^{-4}, 10^{-3}, and 6×10^{-3} respectively. The subjective performance of the reconstructed images by CB-BPGC is also better than those of JPEG2000.

Although the proposed CB-BPGC coder outperforms JPEG2000 on both the compression ratio and the error resilient performance, there are still several issues to be explored in the future.

First of all, as the distribution of the LL subband code block coefficients is not near Laplacian, the current CB-BPGC block coding algorithm performs not very good on those code blocks, thus significantly affecting the lossy compression performance. Better methods to encode the LL subband coefficients and also the sign bits should be further explored in order to improve lossy compression performance.

Second, since complexity in some applications is as important as the compression performance, it is possible to apply simpler neighborhood context modeling techniques in CB-BPGC to reduce complexity. The current neighborhood significant state context modeling process is the most time-consuming part in CB-BPGC.

Additionally, the error resilient performance in CB-BPGC may also be improved by including an estimation process which estimates the lost bits from the probability codebooks when the bitstream is corrupted by channel errors.

BIBLIOGRAPHY

[1] J. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients”, *IEEE Trans. Signal Processing*, vol. 41, pp. 3445-3462, 1993.

[2] A. Said, W. Pearlman, “A new, fast and efficient image codec based on set partitioning in hierarchical trees”, *IEEE Trans. Circuits Syst. Video Technol.*, vol.6, pp.243-250, 1996.

[3] E. Ordentlich, M. Weinberger, G. Seroussi, “A low-complexity modeling approach for embedded coding of wavelet coefficients”, *Proc. DCC’98*, Snowbird, March 1998.

[4] J. Li and S. Lei, “An embedded still image coder with rate-distortion optimization”, *IEEE Trans. Image Processing*, vol. 8, pp. 913-924, July, 1999.

[5] M. J. Weinberger and et at, “The LOCO-1 lossless image compression algorithm: principles and standardization into JPEG-LS”, *IEEE Tran. Image Processing*, vol. 9, pp. 1309-1324, Aug. 2000.

[6] D. Taubman, “High performance scalable image compression with EBCOT”, *IEEE Trans. Image Processing*, vol.9, pp.1158-1170, 2000.

[7] D. Taubman, M. W.Marcellin, *JPEG2000 Image Compression Fundamentals, Standard and Practice*, Kluwer Academic Publishers, Boston / Dordrecht / London, 2002.

[8] ISO/IEC 15444-4:2000 Information technology-*JPEG2000 image coding system-Part 1: Core coding system*, 2000.

[9] ISO/IEC 14492 and ITU-T Recommendation T. 88. JBIG2 bi-level image compression standard, 2000.

[10] Diego Santa-Cruz, Touradj Ebrahimi, “An analytical study of JPEG2000 functionalities”, *IEEE Int. Conf. on Image Processing*, vol. 2, pp. 49-52, 2000.

[11] http://jj2000.epfl.ch/

[12] A. Islam, W. A. Pearlman, “Set partitioning sub-block coding (SPECK)”, ISO/IEC/JTC1/SC29, WG1 N1191, July, 1999.

[13] W. A. Pearlman, et al. “Efficient, low-complexity image coding with a set-partitioning embedded block coder”, *IEEE Tran. on Circuits and Systems for Video Tech.*, vol. 14, no.11, Nov. 2004.

[14] S. T, Hsiang, “Highly scalable subband/wavelet image and video coding”, Ph.D. dissertation, Electrical, Computer and Systems Engineering Dept., Rensselaer Polytechnic Inst., NY, 2002.

[15] S. T. Hsiang, J. W. Woods, “Embedded image coding using zeroblocks of subband/wavelet coefficients and context modeling”, *IEEE Int. Conf. Circuits and Systems (ISCAS)*, vol.3, pp. 662-665, May, 2000.

[16] K. Peng, J. C. Kieffer, “Embedded image compression based on wavelet pixel classification and sorting”, *IEEE Trans. Image Processing*, vol. 13, no. 8, pp. 1011-1017, Aug. 2004.

[17] C. E. Shannon, “A mathematical theory of communication”, *Bell Sys. Tech. Journal*, vol. 27, pp. 379-423 and 623-656, July and October, 1948

[18] I. H. Witten, R. M. Neal, and J. G. Cleary, “Arithmetic coding for data compression”, *Commun. ACM*, vol. 30, pp. 520-540, June 1987.

[19] G. G. Langdon, J. Rissanen, “Compression of black-white images with arithmetic coding”, *IEEE Trans. Communications*, vol. 29, pp. 858-867, 1981.

[20] A. Graps, “An introduction to wavelets”, *IEEE computational science and engineering*, vol. 2, no. 2, Jun. 1995.

[21] http://www.wavelet.org/tutorial/

[22] K. Sayood, *Introduction to data compression*, Morgan Kaufmann Publishers, San Francisco, CA, 2000.

[23] David F. Walnut, *An introduction to wavelet analysis*, Birkhauser, Boston, 2002.

[24] I. H. Witten, Radford M. Neal, and J. G. Cleary, "Arithmetic coding for data compression", *Communication of the ACM*, vol. 30, no. 6, pp. 520-540, 1987.

[25] R.Yu, C.C.Ko, S.Rahardja, X.Lin, "Bit-plane Golomb coding for sources with Laplacian distributions", *IEEE Int. Conf. Accoustics, Speech, and Singal Processing*, 2003.

[26] R.Yu, X.Lin, S.Rahardja, H.Huang, "Technical Description of I2R's Proposal for MPEG-4 Audio Scalable Lossless Coding (SLS): Advanced Audio Zip (AAZ)", ISO/IEC JTC1/SC29/WG11, M10035, Oct. 2003.

[27] R.Yu, X.Lin, S.Rahardja, H.Huang, "Proposed core experiment for improving coding efficiency in MPEG-4 audio scalable coding", ISO/IEC JTC1/SC29/WG11, M10683, Mar. 2004.

[28] J. Li and R. M. Gray, "Context-based multiscale classification of document images using wavelet coefficient distributions," *IEEE Trans. Image Processing*, vol. 9, pp.1604-1616, Sept. 2000.

[29] M. Vetterli, J. Kovacevic, *Wavelets and subband coding*, Prentice Hall Inc., 1995.

[30] C. Lian, K. Chen, H. Chen, L. Chen, "Analysis and architecture design of block-coding engine for EBCOT in JPEG2000", *IEEE Trans. Circuits Syst. Video Technol.*, vol.13, pp.219-230, 2003.

[31] A.said, “Comparative analysis of arithmetic coding computational complexity” *HP Labs Tech. Reports*, HPL-2004-75, 2004.

[32] A.said, “Introduction to arithmetic coding theory and practice”, *HP Labs Tech. Reports*, HPL-2004-76, 2004.

[33] David W. Redmill, Nick G. Kingsbury, “The EREC: an error-resilent techniques for coding variable-length blocks of data”, *IEEE Trans. Image Processing*, vol. 5, no. 4, pp. 565-574, Apr. 1996.

[34] I. moccagatta, et al, “Error-resilient coding in JPEG2000 and MPEG-4”, *IEEE Journ. Selected areas in Communications*, vol. 18, no, 6, Jun. 2000.

[35] Yao Wang and Qin-Fan Zhu, “Error control and concealment for video communication: a review”, *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974-997, May 1998

[36] Jiangtao Wen, John Villasenor, “Reversible Variable Length Codes for Efficient and Robust Image and Video Coding”, *IEEE Proc. Data Compression Conference*, pp. 471-480, Mar. 1998

[37] ISO/IEC JTC1/SC29/WG11/N3908, MPEG-4 video verification model version 18.0, Jan. 2001.

[38] Te-Chung Yang, et al, “Error resilient image coding using low overhead entropy coder and subband dependency”, *IEEE Int. Sym. Circuits and Systems* (ISCAS), 1999.

[39] JPEG2000 image coding system - Part 11: Wireless JPEG2000 - Committee Draft, ISO/IEC JTC1/SC29/WG1 N3386, 2004.

[40] C. Boyd, J. Cleary, S. Irvine, I. Rinsma-Melchert, I. Witten, “Integrating error detection into arithmetic coding”, *IEEE Trans. Commun.*, vol. 45, no. 1, pp. 1-3, Jan. 1997.

[41] Pei-Jun Lee, Liang-Gee Chen, “Bit-plane error recovery via cross subband for image transmission in JPEG2000”, *IEEE Int. Conf. on Multimedia and Expo,* vol.1, pp.149-152, August 2002.

[42] Liu Jieyu, et al, “An efficient error concealment method for JPEG2000 image transmission”, *IEEE Int. Conf. Accoustics, Speech, and Singal Processing*, 2004.

[43] Shuiming Ye, Qibin Sun, Ee-Chien Chang, “Edge directed filter based error concealment for wavelet based image”, *IEEE Int. Conf. on Image Processing*, 2004.

[44] A. Bilgin, Z. Wu, M.W.Marcellin, “Decompression of corrupt JPEG2000 codestreams”, *Proc. Data Compression Conference 2003*, pp 123-132.