

JOINT SOURCE/CHANNEL CODING FOR JPEG2000

by

Lingling Pu

A Dissertation Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
In Partial Fulfillment of the Requirements
For the Degree of
DOCTOR OF PHILOSOPHY
In the Graduate College
THE UNIVERSITY OF ARIZONA

2 0 0 7

FINAL EXAMINING COMMITTEE APPROVAL FORM

As members of the Dissertation Committee, we verify that we have read the dissertation prepared by Lingling Pu
entitled Joint Source/Channel Coding For JPEG2000
and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy

_____ Date: May 8, 2007

Michael W. Marcellin

_____ Date: May 8, 2007

Bane V. Vasic

_____ Date: May 8, 2007

Ivan Djordjevic

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College. I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

_____ Date: May 8, 2007

Dissertation Co-Director: Michael W. Marcellin

_____ Date: May 8, 2007

Dissertation Co-Director: Bane V. Vasic

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: _____
Lingling Pu

ACKNOWLEDGEMENTS

First of all, I would like to thank my thesis advisor, Professor Michael W. Marcellin for his advice, insights and support, and also for giving me the opportunity to work in such an exciting field as joint source/channel coding. Throughout these years, I have learned from him the methods of doing scientific research, the techniques of writing papers, and many other things. This work certainly would not have been possible without his guidance and persistent encouragement.

I would like to express my deep appreciation to my co-advisor, Professor Bane Vasic, for his guidance and insightful discussions. I am fortunate to have had the opportunity to work with him.

My thanks are also due to Professor Ivan Djordjevic, for his help, collaboration and accepting to be my dissertation committee member.

I appreciate the help and friendship of the group members in the Signal Processing and Coding Laboratory. In particular, Dr. Ali Bilgin has always been a huge help in times of need throughout my doctoral study.

I would like to acknowledge the essential work of many other people in the Department of Electrical and Computer Engineering. In particular, Ms. Tami Whelan has given me a lot of help during these years.

Finally, I would like to express my profound gratitude to my parents and brother, whose love and support will always be the most important part of my life; also to my husband, for his love, patience and understanding.

*To my husband,
Peng*

TABLE OF CONTENTS

LIST OF FIGURES	8
LIST OF TABLES	9
ABSTRACT	10
CHAPTER 1 INTRODUCTION	12
1.1 Motivation	12
1.2 Organization and Contributions	15
CHAPTER 2 BACKGROUND	17
2.1 JPEG2000 Source Coding	17
2.2 Channel Coding	22
2.2.1 LDPC Codes	23
CHAPTER 3 LDPC-BASED ITERATIVE JOINT DECODING FOR JPEG2000	25
3.1 Introduction	25
3.2 Error Resilience of JPEG2000	27
3.3 Iterative Joint Source-Channel Decoding of JPEG2000 Codestreams	28
3.4 Experimental Results	32
3.4.1 Ideal Detection of Error Locations	33
3.4.2 Undetected and Mis-detected Source Decoding Errors	38
3.5 Conclusion	41
CHAPTER 4 UNEQUAL ERROR PROTECTION AND PROGRESSIVE DECODING	45
4.1 Introduction	45
4.2 Plotkin Construction of UEP Codes	49
4.2.1 Review of Construction and Decoding	49
4.2.2 Improved Order of Decoding	54
4.2.3 Adjustable Component Code Lengths	55
4.3 A Multi-Stage Decoding Schedule	56
4.4 Experimental Results	58
4.4.1 UEP	59
4.4.2 Application to JPEG2000 Image Transmission	60
4.4.3 Application to Video Transmission	70
4.5 Conclusions	71

TABLE OF CONTENTS — *Continued*

CHAPTER 5 JOINT RATE ALLOCATION IN MULTICHANNEL SYSTEMS	73
5.1 Introduction	73
5.2 Optimization Problem	78
5.3 Applications to Image Transmission	83
5.3.1 Source and Channel Coding	84
5.3.2 Turbo Codes	85
5.3.3 Independent Parallel Gaussian Channels	88
5.3.4 Plastic Optical Fiber Channel	91
5.4 Conclusion	95
5.5 Proofs	96
5.5.1 Proof of Lemma 1	96
5.5.2 Proof of Lemma 2	96
5.5.3 Proof of Corollary 1	96
5.5.4 Proof of Corollary 2	97
REFERENCES	99

LIST OF FIGURES

2.1	A simple JPEG2000 encoder.	18
2.2	Tanner graph of an LDPC code example.	24
3.1	CDF of Δ PSNR for channel SNR = 4.2 dB.	36
3.2	PSNR vs. iteration number for channel SNR=4.3 dB, FB: with feedback; no FB: without feedback.	37
4.1	BER of different components in binary-input AWGN Channels.	60
4.2	BER of UEP code and individual component codes.	61
4.3	Average PSNR vs. number of received packets for BSC $\epsilon = 0.03$ at 0.252 bpp.	69
5.1	A Turbo encoder and decoder diagram.	86
5.2	Packet error rates in AWGN channel.	88
5.3	Block diagram of an OFDM system.	92

LIST OF TABLES

3.1	Average PSNR for 1.0 bpp Lenna, AWGN channel, and Code C_1 with ideal error detection.	33
3.2	Statistics for channel SNR = 4.3 dB, codeblock size 32×32	39
3.3	Average PSNR for 1.0 bpp Lenna, codeblock size 32×32 , and Code C_1	40
3.4	Average PSNR for 1.0 bpp Goldhill, codeblock size 32×32 , and Code C_1	41
3.5	Average PSNR for 0.25 bpp Lenna, AWGN channel, and Code C_1	42
3.6	Average PSNR for 1.0 bpp Lenna, codeblock size 32×32 , and Code C_2	43
4.1	Average PSNR evolution along the scheduling chain for BSC $\epsilon = 0.03$ at 0.252 bpp, EF – error free case.	67
4.2	Average PSNR from UEP and EEP for BSC channels at 0.252 bpp.	70
4.3	Average PSNR evolution along the scheduling chain for AWGN=2.8 dB, EEP – Equal error protection.	71
5.1	Average MSE and PSNR for parallel Gaussian channels.	91
5.2	Average MSE and PSNR for 1.0 bpp/source and GI-POF channel.	94

ABSTRACT

In today's world, demands of digital multimedia services are growing tremendously, together with the development of new communication technologies and investigation of new transmission media. Two common problems encountered in multimedia services are unreliable transmission channels and limited resources. This dissertation investigates advanced source coding and error control techniques, and is dedicated to designing joint source-channel coding schemes for robust image/video transmission. Error resilience properties of JPEG2000 codestreams are investigated first, and an LDPC-based joint iterative decoding scheme is proposed. Next, a progressive decoding method is presented for still and motion image transmission. The underlying channel codes are created using a Plotkin construction and offer the novel ability of using one long channel codeword to protect an entire image, yet still allowing progressive decoding. Progressive quality improvements occur in two ways: the first is the usual progressive refinement, where image quality is improved as more data are received; the second is that residual error rates of earlier received data are reduced as more data are received. Finally, multichannel systems are studied and an optimal rate allocation algorithm is proposed for parallel transmission of scalable images in

multichannel systems. The proposed algorithm selects a subchannel as well as a channel code rate for each packet, based on the signal-to-noise ratios (SNR) of the subchannels. The resulting scheme provides unequal error protection of source bits and significant gains are obtained over equal error protection (EEP) schemes. An application of the proposed algorithm to JPEG2000 transmission shows the advantages of exploiting differences in SNRs between subchannels. Multiplexing of multiple sources is also considered, and additional gains are achieved by exploiting information diversity among the sources.

CHAPTER 1

INTRODUCTION

1.1 Motivation

In today's world, demands of digital multimedia services are growing tremendously, together with the development of new communication technologies and investigation of new transmission media. Such increasing demands can be seen, as an example, from the mobile phone services provided nowadays, including multimedia messaging, digital cameras, MP3 players, etc. IPTV (Internet Protocol Television) is another example, which is becoming more and more prevalent. There are over a thousand free IPTV channels available at the time of this writing, some of which also support HDTV format.

These multimedia services provide great convenience, but transmission over unreliable channels is sometimes inevitable. Thus, error control is necessary in such systems. Another common system requirement is source coding, i.e., data compression. This is due to the fact that communication systems have limited resources such as bandwidth or power. Thus, a successful communication system is desirable to be able to transmit as much information as possible, with an error

rate as low as possible. The two requirements can be satisfied separately. Forward error correction (FEC) codes and ARQ (automatic repeat request) can be used for error control, while source coding can efficiently remove the redundancy residing in multimedia sources. It is proved by Shannon [1] that source coding and channel coding can be performed separately and sequentially without loss of optimality. He presented a theorem stating that a source can be reliably transmitted through a channel, as long as the source entropy H does not exceed the channel capacity C . In light of the theorem, an ideal source coder can first try to reduce the source rate to the minimum value of H , then an ideal channel coder can be applied to achieve arbitrarily small error rate. The two coders are optimized without knowledge of each other's statistics. This theorem is known as the separation theorem.

Separation simplifies system design, but the conditions of the theorem often do not hold in practice. With complexity and delay constraints, ideal channel coding can not be achieved since it generally requires infinite length code words. The theorem does not hold for non-ergodic and multi-user channels. When the channel has varying qualities, the optimality can not be guaranteed. In order to accommodate the above, a joint source-channel coding (JSCC) approach is needed. How to utilize the knowledge of the other one to assist encoding/decoding for each coder is a key point in a joint design, at the same time, efficient source coding and high performance error control techniques are also crucial to the end-to-end

performance.

For source coding, wavelet-based coders have become more and more prevalent, because of the scalability and competitive rate-distortion performance. Examples of such coders include EZW (embedded zerotree wavelet) [2], SPIHT (set partitioning in hierarchical trees) [3], and JPEG2000 [4]. JPEG2000 is the latest international standard for still image compression. It not only improves compression performance over precedent standards like JPEG, but also provides a rich feature set including scalability and editability. JPEG2000 can serve well in many applications, such as digital cameras, 3G mobile phones, client/server communication, medical imagery, digital cinema, and digital libraries/archives, etc. This dissertation will focus on JPEG2000-coded sources.

For error control, development of capacity-achieving channel codes and iterative decoding algorithms makes FEC-based approaches an attractive research topic in JSCC for multimedia transmission. Turbo codes [5] and low density parity check (LDPC) codes [6] [7] are two excellent examples of the development. Another popular method for error control is to investigate error resilience properties of source coders, which could effectively prevent error propagation together with some error detection mechanism.

The desire to combine the advanced techniques mentioned above and design

a robust and efficient multimedia communication system motivates this dissertation.

1.2 Organization and Contributions

This dissertation focuses on joint source-channel coding designs for image/video transmission over noisy channels.

Chapter 2 introduces relevant background knowledge of source and channel coding techniques used in this dissertation.

In Chapter 3, a framework is proposed for iterative joint source-channel decoding of JPEG2000 codestreams. At the encoder, JPEG2000 is used to perform source coding with certain error resilience (ER) modes, and LDPC codes are used to perform channel coding. During decoding, the source decoder uses the ER modes to identify corrupt sections of the codestream, and provides this information to the channel decoder. Decoding is carried out jointly in an iterative fashion. The resulting scheme requires fewer iterations than separate decoding and improves overall system performance.

In Chapter 4, an unequal error protection (UEP) scheme is presented for still and motion image transmission. The underlying channel codes are created using a Plotkin construction and offer the novel ability of using one long channel

codeword to protect an entire image, yet still allowing progressive decoding. Progressive quality improvements occur in two ways: the first is the usual progressive refinement, where image quality is improved as more data are received; the second is that residual error rates of earlier received data are reduced as more data are received.

In Chapter 5, a novel rate-optimal rate allocation algorithm is proposed for parallel transmission of scalable images in multichannel systems. Scalable images are transmitted via fixed-length packets. The proposed algorithm selects a subchannel as well as a channel code rate for each packet, based on the signal-to-noise ratios (SNR) of the subchannels. The resulting scheme provides unequal error protection of source bits and significant gains are obtained over equal error protection (EEP) schemes. An application of the proposed algorithm to JPEG2000 transmission shows the advantages of exploiting differences in SNRs between subchannels. Multiplexing of multiple sources is also considered, and additional gains are achieved by exploiting information diversity among the sources.

CHAPTER 2

BACKGROUND

This chapter introduces background information that is essential to treatments of later chapters. Section 2.1 gives basic mechanism of JPEG2000, and Section 2.2 discusses the issues of channel coding, particularly the structures and decoding algorithms of LDPC.

2.1 JPEG2000 Source Coding

A description of JPEG2000 encoding is provided below, aimed to introduce the key mechanisms of the JPEG2000 standard, and some of the rich feature set provided thereby. Those features are used in the joint designs described later. The description is rather brief, and readers are referred to [4] for more details.

Figure 2.1 shows a simple diagram of a JPEG2000 encoder. An input image is first optionally divided into non-overlapping rectangular regions, called tiles. Each tile will be transformed and encoded independently. The main purposes of tiles are for memory efficient implementation and spatial random access. Note that there are other ways in JPEG2000 to achieve the above purposes (as will be discussed later).

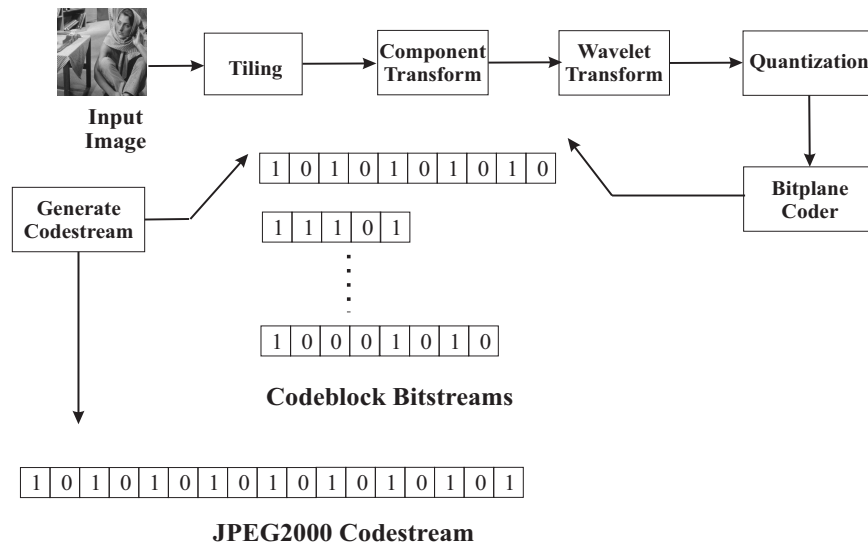


Figure 2.1: A simple JPEG2000 encoder.

A disadvantage of tiles is they may produce “blocking” artifacts, especially at low bit rates. The use of tiles can be avoided by treating the entire image as a single tile. If the image has multiple components, either a reversible or irreversible color transform can be performed on the components. The color transform can decorrelate the components, increasing compression efficiency. Image samples within a tile from one component form a *tile-component*, which is then transformed by discrete wavelet transform (DWT).

There are several geometric structures based on the collection of wavelet coefficients after DWT. After one level of DWT, the coefficients are grouped into four bands, called *subbands*. Each subband is associated with a certain frequency range, among which the low-low (LL) subband can be viewed as a reduced resolution version of the tile-component. This observation implicates an important feature

of JPEG2000, that it provides reduced resolution imagery within its codestream. Multiple levels of DWT can be further performed to the LL subband.

Each resolution of a tile-component is partitioned into *precincts*. Precincts behave much like tiles, but in the wavelet domain. Like tiles, precincts benefit low memory implementations and spatial random access. An advantage over tiles is that precincts won't cause "block" artifacts since they are in wavelet domain.

The smallest geometric structure of JPEG2000 is the *codeblock*. Codeblocks can be viewed as partitions of precincts. Quantization and bitplane coding are performed on codeblocks. As a consequence, codeblocks enable low memory implementations too, since only coefficients from one codeblock need to be buffered before quantization and bitplane coding. Codeblocks can also help error resilience because of the independent encoding of each codeblock. More details on error resilience will be discussed in Chapter 3.

Quantization is a key step in data compression, for it reduces the precision of the data. In JPEG2000 (Part I), deadzone uniform scalar quantization is used. This enables the generation of embedded codestreams and quality progression. An embedded codestream can successively refine the reconstructed data as more data of the codestream are received and decoded. More rigorously speaking, every L -bit prefix of the embedded codestream is itself an efficient representation of the source.

It is a very desirable feature provided by JPEG2000.

The quantized coefficients are entropy coded. This is done by the bitplane coder. Bitplanes are binary arrays taken from the integer sign-magnitude representations of quantized coefficients. For example, the first such array contains the most significant bit (MSB) of all the magnitudes. Starting from the most significant bits and to less significant bits, each bitplane of the codeblock is encoded in three *coding passes*. The three passes are called *significance propagation pass*, *magnitude refinement pass* and *cleanup pass*, respectively. A context dependent, binary arithmetic coder, called the MQ-coder, is employed to encode the bits collected by the coding passes.

The last step indicated by Figure 2.1 is the construction of the compressed codestream. The fundamental unit of the construction is *packet*. A packet contains some number of coding passes from each codeblock in one precinct of one tile-component. The syntax of a packet includes a packet header and packet body. The packet header contains critical information so that if it is lost, the packet can not be decoded. Packets provide a convenient way to achieve quality scalability. The first packet of a particular precinct contains the most important coding passes (a collection of some number of MSBs). Later packets provide quality increments. Packets can be reordered in the JPEG2000 codestream. Four aspects of progression can be achieved: quality, resolution, position, component. This gives the encoder

as well as image servers a high degree of freedom.

Another structure of quality scalability is *layers*. A layer is a collection of packets from one tile. More specifically, a layer collects one packet from each precinct of each resolution of each tile-component of one tile.

A JPEG2000 codestream is constructed from the compressed packets. Besides the compressed packet data, a main header is formed at the beginning of a codestream, which signals information necessary for decompression, such as image size, tile size, codeblock size and quantization step sizes, etc. As with packet headers, if the main header is corrupted, the decompression may not be performed successfully. As shall be seen in later chapters, main header and packet headers are always treated with particular care.

Before leaving the high level description of the encoding procedure, a rate control technique for JPEG2000 is discussed. Distortion is commonly measured as mean squared error (MSE). For a stationary random process, the distortion-rate function is known to be convex [8]. Practical compression systems exhibit the same characteristic. When a codeblock is encoded, it generates an embedded bit-stream. Feasible truncation points are expected to lie close to the convex rate-distortion function for the corresponding source. Optimal truncation is taken care of by a post-compression rate-distortion optimization (PCRD-opt) algorithm. A figure of

merit used in PCRD-opt is the *distortion-rate slope*, which is computed as the ratio $\Delta MSE/\Delta R$. ΔMSE represents the amount MSE is reduced by increasing the bit-stream by a length of ΔR . The coding pass end points form a natural set of truncation points. The definition of the three coding passes in JPEG2000 and the way they are encoded are designed to separate information in each bit plane into multiple coding passes with decreasing rate-distortion slopes, so that a good approximation of convex rate-distortion characteristic is achieved. As a result, coding passes with larger slopes should be included earlier than those with smaller slopes. During encoding, a rate-distortion slope for each coding pass is computed. Packets are formed by including coding passes from largest slope to smallest, and layers are formed when the selected coding passes meet target bit rates for corresponding layers.

2.2 Channel Coding

This section introduces some of the channel coding techniques commonly used in the literature of JSCC. Particularly, LDPC codes are discussed. These codes exhibit near capacity-achieving performances, and are used in the treatments discussed in later chapters.

2.2.1 LDPC Codes

LDPC codes were first proposed by Gallager [7] in 1960, but it is not until in the mid-1990's that their advantages were rediscovered by MacKay and others [6], [9], [10].

LDPC codes are a class of linear block codes, which have sparse parity check matrices. The word “sparse” means a low density of the element “1” in the binary parity check matrix. For illustration purposes, a parity check matrix of an LDPC code could be

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}. \quad (2.1)$$

Tanner [11] showed an effective way to represent the structure of LDPC codes, using a bipartite graph, referred as a Tanner graph. A Tanner graph contains two classes of nodes, usually called *variable nodes* and *check nodes*. Only nodes from different classes can be connected by edges. The bipartite graph of the example LDPC code is shown in Figure 2.2. Whenever a matrix element h_{ji} in H is 1, there is a corresponding edge in the Tanner graph that connects check node c_j and variable node v_i . The example shown above is a regular LDPC code, i.e., it has constant column weight w_c and row weight w_r . For irregular LDPC codes, w_c and w_r vary among different columns and rows. In that case, degree distribution polynomials can be used to specify the weights. For a legal codeword c , the relation holds: $cH^T = 0$. Each row of H can be viewed as a check equation. The first row of H

in the example constrains the variable nodes (bits) so that the following equation needs to be satisfied: $v_0 + v_1 + v_2 + v_3 = 0$. The sum is a mod-2 sum.

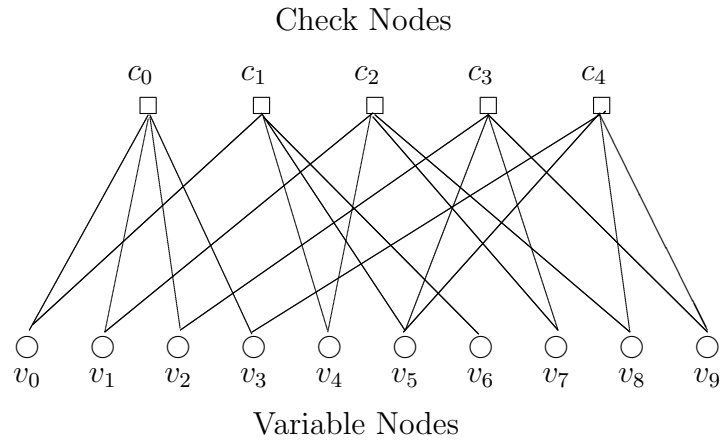


Figure 2.2: Tanner graph of an LDPC code example.

There are several iterative decoding algorithms of LDPC codes, such as sum-product algorithm, min-sum algorithm and belief propagation algorithm. The decoder computes the log-likelihood ratio (LLR) of variable nodes

$$L(c_i) \triangleq \log \left(\frac{\Pr(c_i = 0|y)}{\Pr(c_i = 1|y)} \right)$$

by sending information back and forth between variable nodes and check nodes.

CHAPTER 3

LDPC-BASED ITERATIVE JOINT DECODING FOR JPEG2000

3.1 Introduction

Variable length codes (VLC) are widely used in practical image or video compression systems, such as JPEG, JPEG2000, MPEG and H.264. VLC are sensitive to synchronization errors. In the literature, joint source channel coding techniques are investigated to achieve reliable transmission of compressed codestreams in noisy channels.

In Kozintsev et al. [12], a rate-compatible punctured convolutional code (RCPC) is concatenated with an arithmetic coder. A forbidden symbol never encoded by the arithmetic coder is assigned a nonzero probability nonetheless. Whenever the forbidden symbol is decoded, error occurrence is detected. Steingrimsson et al. [13] propose a signal space detector used along with FEC codes to enhance the decoding process for the DVD channel. Peng et al. [14] apply Turbo codes of rate 0.5 to compressed images/video coded by different source coding schemes, such as vector quantization, JPEG and MPEG. Redundant source information, or some unique structure in these source codes, are used by the channel decoder. Guionnet

and Guillemot [15] propose error-resilient soft decoding of arithmetic codes. They also describe a joint scheme combining arithmetic codes and convolutional channel codes, but state that iterations bring no significant gain. There is also intensive work in designing unequal error protection schemes for embedded sources. Banister et al. [16] propose an efficient system to protect JPEG2000 coded images using Turbo codes. The Viterbi algorithm was used to find an optimal rate allocation between source and channel codes, minimizing the distortion in reconstructed images. Lan et al. [17] consider the design of punctured irregular repeat-accumulate codes, and their application to a UEP scheme for scalable image and video coders. Pan et al. [18] investigate the design of rate-compatible LDPC codes, and use the codes to protect JPEG2000 coded images. Zhong and Havlicek [19] use LDPC codes and Reed-Solomon codes to protect baseline JPEG images.

In this chapter, we propose a joint source-channel decoding scheme based on JPEG2000 [4] and LDPC channel codes. This approach focuses on combining error resilience properties of JPEG2000 coding and error correction capabilities of LDPC codes. More specifically, JPEG2000 is used to perform source coding with certain error resilience (ER) modes, and LDPC codes are used to perform channel coding. During decoding, the source decoder uses the ER modes to identify corrupt sections of the codestream and provides this information to the channel decoder. The decoding is carried out jointly in an iterative fashion. The resulting scheme can

be thought of as a concatenated error control scheme. The source coder is serving as an outer code, providing error detection to an inner LDPC error correction code, similar to the schemes in [12], [14].

Section 3.2 introduces error resilience mechanisms of JPEG2000. An iterative joint source channel decoding scheme of JPEG2000 is discussed in Section 3.3. Section 3.4 lists experimental results, and conclusions are included in Section 3.5.

3.2 Error Resilience of JPEG2000

Data partitioning is a common practice used for error resilience in data compression systems. In JPEG2000, codestreams are partitioned into a series of hierarchical structures, starting with a main header. The main header contains crucial information, such as image and tile size, number of components, transform levels and quantization step sizes, etc. A sequence of tile-streams follows the main header. Each tile-stream consists of a tile header and a sequence of packets. Each packet contains a packet header and a packet body.

Headers are of great importance. If the main header is not received correctly, the decoder may not be able to decode the codestream. If a tile header is not received correctly, that tile might be discarded. If a packet header is lost, the current and following packets of the corresponding precinct will be discarded. There is a mechanism in JPEG2000 to extract packet headers and store them in the main

header. This is referred to as *packed packet headers*. This mechanism facilitates UEP design to give strong protection to the crucial information.

Several mode switches in JPEG2000 provide resynchronization capability, including the arithmetic coder switch RESTART. RESTART causes the arithmetic coder to be restarted at the beginning of each coding pass. In this case, each coding pass has a separate arithmetic codeword segment. ERTERM is a mode that enables error detection capability. When the ERTERM switch is used, a predictable termination policy is employed for each codeword segment. A decoder is able to reliably detect when an arithmetic codeword segment is corrupted. If the JPEG2000 codestream is generated using these two mode switches, the decoder can identify when an error has occurred in a given coding pass. When an error occurs in a coding pass, common practice is to discard the current and all future coding passes of the current codeblock. The decoder then starts decoding the first coding pass in the next codeblock. In this way, bit errors do not propagate from one codeblock to the next.

3.3 Iterative Joint Source-Channel Decoding of JPEG2000 Code-streams

In this work, we enable the RESTART and ERTERM switches introduced above, and combine the error correction capability of LDPC codes to design an iterative

joint decoding scheme. Information on which coding passes in each codeblock are decodable is fed back to the LDPC decoder.

To see how this feedback source information can help the channel decoder, examination of the LDPC decoding algorithm is needed. For BPSK modulation, each bit C_i of an LDPC codeword is mapped to symbol x_i , +1 or -1, corresponding to it being 0 or 1. The set of C_i corresponds to the set of variable nodes in a Tanner graph [11]. The Tanner graph also has a set of check nodes defined by the parity check matrix of the code. The transmitted symbol x_i has a received (noisy) value of y_i , and the log *a posteriori* ratio of C_i is

$$\log \frac{\Pr(x_i = +1|y_i)}{\Pr(x_i = -1|y_i)} = \log \frac{\Pr(y_i|x_i = +1) \Pr(x_i = +1)}{\Pr(y_i|x_i = -1) \Pr(x_i = -1)}. \quad (3.1)$$

Assume an additive white Gaussian noise (AWGN) channel and equal *a priori* probabilities. Then, the initial log-likelihood ratio (LLR) of C_i is

$$L(C_i) = \log \frac{\Pr(y_i|x_i = +1)}{\Pr(y_i|x_i = -1)} = 2y_i/\sigma^2. \quad (3.2)$$

The decoding procedure starts from the initial LLRs and messages are passed iteratively between variable nodes and check nodes. At the end of decoding, the LLR for each variable node $L(C_i)$ is calculated and compared to the threshold. More particularly, if $L(C_i) \geq 0$, decision $C_i = 0$ is made; otherwise $C_i = 1$.

In equation (3.2), the *a priori* probabilities of x_i are assumed equal. After source decoding, information on where the first bit error occurs is available. We

employ this extrinsic information in the LDPC decoding algorithm. Specifically, the *a priori* probabilities of x_i are re-defined and calculation of the $L(C_i)$ is modified. The details of our scheme are described in the following paragraphs.

After JPEG2000 encoding of an image, the resulting codestream is sequentially divided into channel codewords. These channel codewords are mapped into channel symbols as $x_i = (-1)^{C_i}$. The received symbols are $y_i = x_i + n$. One iteration of LDPC decoding is performed and the output codestream is then decoded by JPEG2000. The correct coding passes are detected in each channel codeword. This source information is passed to the channel decoder.

First suppose that the source decoder performs perfect error detection. For those bits in correct coding passes, $\Pr(x_i = +1) = 1$, if $L(C_i) \geq 0$ and $\Pr(x_i = -1) = 1$, if $L(C_i) < 0$. These bits are now known to the channel decoder, and they can help decoding other undetermined bits involved in the same check equations. These known bits are assigned LLRs of $L(C_i) = \infty$, if $L(C_i) \geq 0$ and $L(C_i) = -\infty$, if $L(C_i) < 0$. In practice, large values are used instead of infinity.

Now suppose that the source decoder performs correct error-detection with some probability $p_{det} < 1$. For those bits in correct (believed by the source decoder)

coding passes, Equation (3.2) now becomes

$$\begin{aligned}\hat{L}(C_i) &= \log \frac{\Pr(y_i|x_i = +1)}{\Pr(y_i|x_i = -1)} + \log \frac{\Pr(x_i = +1)}{\Pr(x_i = -1)} \\ &= L(C_i) + \text{sgn}\{L(C_i)\} \cdot \log \frac{p_{det}}{1 - p_{det}},\end{aligned}\tag{3.3}$$

where sgn is the sign function. The bits referred to in Equation (3.3) are bits in the LDPC codeword, and they correspond to compressed source bits, which are largely independent.

In summary, we modify the soft values of the variable nodes involved in correct coding passes as:

$$\hat{L}(C_i) = \begin{cases} L(C_i) - t, & \text{if } L(C_i) < 0; \\ L(C_i) + t, & \text{if } L(C_i) \geq 0. \end{cases}\tag{3.4}$$

The weighting factor t is defined as:

$$t = \left| \log \frac{\Pr(x_i = +1)}{\Pr(x_i = -1)} \right|\tag{3.5}$$

It is important to note that coding passes must be treated sequentially. Due to the context dependent arithmetic coding employed in JPEG2000, a coding pass can only be decoded when all the previous coding passes in the same codeblock are correct. Thus the soft values can be modified for all bits within a codeblock, up to but not including those in the first coding pass containing incorrectly decoded bits. After modification of the appropriate soft values for all code blocks, the next

iteration of channel decoding is performed, followed by another round of source decoding. This iterative decoding procedure is repeated until some stopping criterion is met.

3.4 Experimental Results

An important factor that affects overall performance is the codeblock size used during the JPEG2000 compression procedure. We have used the Kakadu V3.3 implementation of JPEG2000 [20]. By default, Kakadu uses 64×64 codeblocks. In addition to this size, we have also tested our scheme using codeblock sizes of 32×32 , 16×16 and 8×8 . Smaller codeblock sizes result in shorter coding passes. This has the negative effect of some reduction in compression efficiency when no channel errors are present. On the other hand, shorter coding passes are preferable when channel errors are present, since they provide better error localization.

In our experiments, two LDPC codes are selected. The first code, C_1 is a regular $(3648, 3135)$ code based on Hermitian unitals with girth 6 [21]. The second code, C_2 is a $(3648, 2748)$ code based on lattice design [22]. Two codes of different structures and rates are chosen to show that the proposed scheme is insensitive to the particular choice of codes. In each case where a noisy channel is employed, 10000 simulations are performed. MSE (Mean Squared Error) values of decoded images are averaged over the simulations, then the average MSE is converted to PSNR. The

Table 3.1: Average PSNR for 1.0 bpp Lenna, AWGN channel, and Code C_1 with ideal error detection.

Channel SNR	Codeblock Size	With FB	No FB	Δ PSNR
4.2 dB	32×32	23.97	22.80	1.17
	16×16	24.55	23.04	1.51
	8×8	30.50	24.07	6.43
4.3 dB	32×32	28.01	26.53	1.48
	16×16	28.91	26.91	2.00
	8×8	34.83	27.85	6.98
4.4 dB	32×32	32.91	31.03	1.88
	16×16	33.09	30.73	2.36
	8×8	37.19	31.67	5.52
4.5 dB	32×32	36.89	34.98	1.91
	16×16	36.72	34.53	2.19
	8×8	37.77	34.96	2.81

512×512 Lenna and Goldhill images, compressed at 1.0 and 0.25 bits/pixel source rates, are used as the test images. The headers of the compressed codestreams are assumed to be protected perfectly in the experiments.

3.4.1 Ideal Detection of Error Locations

We first assume exact information on which coding pass contains the first error within each code block, *i.e.*, error detection is forced to be correct artificially, so that there are no incorrect or mis-detections of corrupt coding passes. The weighting factor is then $t = \infty$. Due to the LDPC decoding algorithm, extremely large values of t yield comparable results to those obtained with moderate values of t . In the experiments, t is chosen to be 5.

Table 3.1 lists the results of our joint decoding method versus the usual separate decoding method when the channel SNR equals 4.2, 4.3, 4.4, and 4.5 dB.

For codeblock sizes of 32×32 , 16×16 and 8×8 , error-free decoding yields PSNR values of 39.80 dB, 39.07 dB, and 37.87 dB, respectively. From this it can be seen that the chosen values of SNR correspond to the region where the channel codes are beginning to fail. In other words, we are operating near (below) the knee in the PSNR vs. channel SNR curves. “With FB” represents the proposed method with feedback while “No FB” represents the separate decoding method. Forty iterations are carried out for each method. Results for more iterations are reported later. The proposed method outperforms separate decoding consistently, effectively extending the knee of the PSNR/SNR curve. On the other hand, if the channel SNR continues to increase, the performance of the proposed method gets close to error-free decoding much faster than separate decoding. We note that in Table 3.1, for a channel SNR of 4.5 dB and a codeblock size of 8×8 , the average PSNR from our joint decoding method (37.77 dB) is close to the error-free PSNR (37.87 dB). This is consistent with the percentage of images decoded perfectly, which is 99.96% from our proposed method. The percentage of perfectly decoded images from separate decoding is only 70.62%.

From the table, we can see the effect of the codeblock size. Δ PSNR increases when codeblock size decreases, indicating that the proposed method is more effective for smaller codeblocks. In this case, better error resiliency dominates the end-to-end performance. It can also be observed that under lower channel SNR

conditions, ΔPSNR is generally larger, consistent with extension in the knee of the curve. However eventually, in the very low channel SNR case, the PSNR results from both methods are extremely low, rapidly approaching zero. Consequently, the ΔPSNR between the two methods must also approach zero. For the smaller codeblock sizes, ΔPSNR continues to increase somewhat longer (compared to the larger codeblock sizes) as channel SNR is decreased, before eventually falling precipitously. It is worth noting that high channel SNR (essentially error-free decoding) results in PSNR values that decrease when codeblock size decreases. On the other hand, low channel SNR results in PSNR values that increase when codeblock size decreases. In between then, there is a region where PSNR is not monotonic in codeblock size.

The distribution of the gains (*i.e.*, ΔPSNRs) between the two methods is another illustrative way to show the performance. Fig. 3.1 shows the Cumulative Distribution Function (CDF) of ΔPSNR for different codeblock sizes. For example, it can be seen in Fig. 3.1 that the proposed method results in PSNR gains larger than 1 dB in roughly 40% of the simulations for a codeblock size of 32×32 , 75% and 96% for codeblock sizes of 16×16 and 8×8 respectively. For gains larger than 5 dB, those percentages are 8%, 20%, and 60% respectively.

In Fig. 3.1, there are cases when ΔPSNR is negative. The joint decoding method corrects more erroneous bits on average. However, different specific bits are decoded incorrectly in each case. This, together with the fact that different source

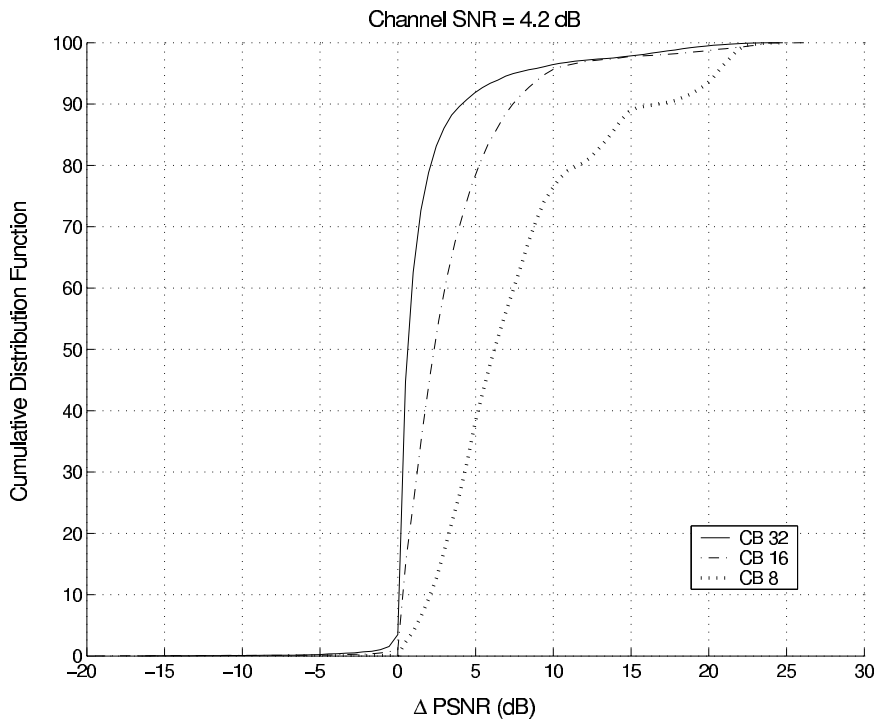


Figure 3.1: CDF of Δ PSNR for channel SNR = 4.2 dB.

bits are of different importance, can explain the occasional negative Δ PSNR values.

The PSNR vs. number of iterations is shown for the two schemes in Fig. 3.2. We note that the proposed scheme is better at every iteration. The complexity of the proposed method is essentially the same as that for the separate scheme. We can see this by comparing the run times of the algorithm components. The average computation time during channel decoding was measured as 6.80 seconds/iteration. The proposed method adds a source decoding operation to each iteration, which was measured as 0.01 seconds/iteration. From this, we conclude that the joint scheme has essentially the same complexity as the separate scheme for software

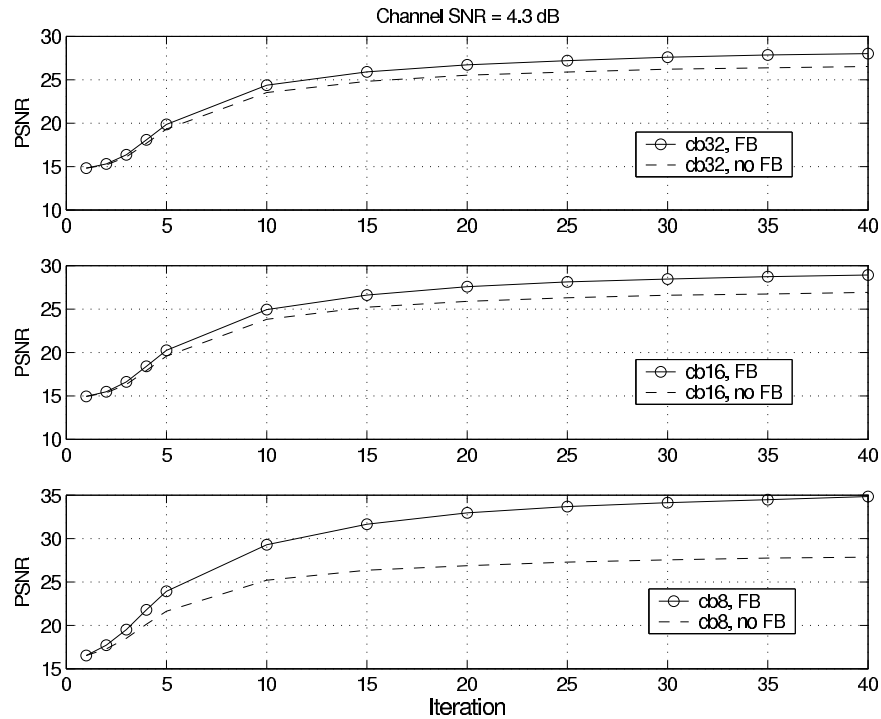


Figure 3.2: PSNR vs. iteration number for channel SNR=4.3 dB, FB: with feedback; no FB: without feedback.

implementation. Hardware implementations may yield differing conclusions. From Fig. 3.2, we can also see that on average, PSNR is monotonic with iteration. This is not necessarily true for each individual realization. In fact, the residual error rate for (separate) LDPC decoding is not always monotonic. It is interesting to note however, that the number of cases when the residual error rate increases is significantly lower (by a factor of at least 5) when joint decoding is performed.

3.4.2 Undetected and Mis-detected Source Decoding Errors

We now consider a more realistic use of the proposed joint decoding method. In what follows, Kakadu is tasked to find the location of the first corrupt coding pass of each codeblock. We begin by studying the reliability of Kakadu in performing this job. To this end, we corrupt Kakadu codestreams with a BSC having error probability ϵ . During decoding of the compressed image, we generate an error report about the position of the first error Kakadu detects within each code block. By comparing the error report with the truth, the statistical results of error-detection performance are found. The value of ϵ is selected as the average residual BER after some number of iterations for the channel code used in the joint decoding scheme. Even though BSC may not be a good model for the residual errors, we use it for its simplicity and to gain some insight into the error detection capabilities of Kakadu.

Table 3.2 lists the results under a channel SNR of 4.3 dB and differing numbers of iterations. The results are based on 50,000 simulations. The entry “success” in the table is the percentage of cases when the first error in a codeblock is detected as occurring in the correct coding pass. “Delay 1” is the percentage of the errors which are detected one coding pass later than their true occurrences. “Delay 2” is the same as “Delay 1” but two coding passes later. Further delays are negligible. “Miss” is the percentage of cases when an existing error goes undetected.

Table 3.2: Statistics for channel SNR = 4.3 dB, codeblock size 32×32 .

Iteration	ϵ	Success	Delay 1	Delay 2	Miss
5	3.19×10^{-3}	98.47%	0.66%	0.55%	0.29%
10	1.42×10^{-3}	98.28%	0.67%	0.60%	0.36%
20	8.10×10^{-4}	98.17%	0.65%	0.61%	0.44%
40	5.81×10^{-4}	98.13%	0.64%	0.60%	0.49%

More than 90% of the “Miss” cases occur when the true location of the error is the last coding pass of a code block.

From these statistics a suitable weighting factor t can be calculated. Suppose Kakadu reports an occurrence of the first error in the j th coding pass. Then from Equation (3.5), the weighting factor for the $(j - 1)$ th coding pass should be calculated as:

$$t_{j-1} = \log \frac{\text{success}}{1 - \text{success}} \quad (3.6)$$

Because of the fairly high reliability of detection, this weighting factor is large. This value of t is computed based only on probability of incorrect detection. However, if a coding pass is marked as error-free while it actually contains one or more errors, dramatic increases in image distortion can result. Thus, values for t chosen strictly based on error probabilities using the analysis above yield poor performance, since this approach does not take into account the MSE penalty for declaring a coding pass to be correctly decoded when in fact it is not. It is desirable to derive an improved expression for t that incorporates MSE of the decoded imagery rather than just residual bit error probability. However such derivation is left for future

Table 3.3: Average PSNR for 1.0 bpp Lenna, codeblock size 32×32 , and Code C_1 .

Channel SNR	With FB	Without FB	Δ PSNR	Δ PSNR Ideal
4.2 (dB)	23.48	22.80	0.68	1.17
4.3 (dB)	27.31	26.53	0.78	1.48
4.4 (dB)	32.05	31.03	1.02	1.88
4.5 (dB)	35.89	34.98	0.91	1.91

work. In this work, we choose conservative weights for coding passes where detection errors are most probable. In our experiments, we have found that values for t of 0.5 and 0.2 work well for the $(j - 2)$ th and $(j - 1)$ th coding passes, respectively. As in the error free case, we use $t = 5$ for the $(j - i)$ th coding pass for all $i > 2$.

Results of experiments employing these values of t and a codeblock size of 32×32 , are listed in Table 3.3. The gains obtained have the same tendency as before. But, as expected, gains are not as large as those in the ideal case, when perfect error detection was assumed. The ideal results from Table 3.1 are repeated in Table 3.3 for comparison. With more complex weighting schemes as hinted at above, or additional error detection schemes, the performance of practical joint decoding can be closer to the ideal case.

Results for other images and bitrates are similar. To this end, the proposed scheme was also tested for the Goldhill image using code C_1 . The results are listed in Table 3.4 for 1 bit/pixel. The error-free PSNR for the Goldhill image is 35.95 dB. Table 3.5 lists the PSNR values for the Lena image at 0.25 bit/pixel, with

Table 3.4: Average PSNR for 1.0 bpp Goldhill, codeblock size 32×32 , and Code C_1 .

Channel SNR	With FB	Without FB	Δ PSNR
4.2 (dB)	23.16	22.25	0.91
4.3 (dB)	26.99	25.92	1.07
4.4 (dB)	30.83	29.80	1.03
4.5 (dB)	34.18	33.39	0.79

codeblock sizes of 32×32 and 16×16 . A maximum of 100 iterations are used in this experiment. The PSNR numbers in parentheses are the results at 40 iterations. As expected the results are improved for both methods, but the trends are preserved. At channel SNR 4.4 and 4.5 dB, the gains at 40 iterations are larger than the gains at 100 iterations, while at 4.2 and 4.3 dB gains at 100 iterations are larger. This is caused by faster convergence of the joint scheme at higher channel SNRs than at lower channel SNRs. Nevertheless, the proposed joint decoding scheme with more iterations always yields the best results in PSNR. To show the robustness of the proposed method over different channel codes, results for the Lenna image using code C_2 are shown in Table 3.6. As expected, the operational channel SNR range in which gains are obtained is dependent on the channel code. However, the SNR range of interest occurs always at the knee of the PSNR/SNR curve.

3.5 Conclusion

From the experimental results, it is clear that the proposed joint iterative decoding method can improve overall system performance. The proposed method can be

Table 3.5: Average PSNR for 0.25 bpp Lenna, AWGN channel, and Code C₁.

SNR	Codeblock	With FB	No FB	Δ PSNR
4.2 dB	32×32	23.52(23.12)	22.91(22.57)	0.61(0.55)
	16×16	23.88(23.48)	23.09(22.75)	0.79(0.73)
4.3 dB	32×32	27.22(26.61)	26.37(25.83)	0.85(0.78)
	16×16	27.69(27.06)	26.64(26.19)	1.05(0.87)
4.4 dB	32×32	30.85(30.31)	30.18(29.61)	0.67(0.71)
	16×16	30.74(30.25)	30.07(29.54)	0.67(0.71)
4.5 dB	32×32	32.90(32.62)	32.63(32.27)	0.27(0.35)
	16×16	32.22(31.99)	31.91(31.50)	0.31(0.49)

Table 3.6: Average PSNR for 1.0 bpp Lenna, codeblock size 32×32 , and Code C_2 .

Channel SNR	With FB	Without FB	Δ PSNR
2.6 (dB)	23.79	23.00	0.79
2.8 (dB)	31.55	29.78	1.77
3.0 (dB)	36.02	35.14	0.88
3.2 (dB)	38.51	37.77	0.74

seen to extend the knee of the PSNR vs. channel SNR curve, effectively extending the usefulness of any channel code to lower channel SNR. Certain parameters in the proposed method are chosen empirically. However, the parameters chosen are robust over a wide variety of images and channel codes. Different channel codes can be chosen to suit the channel condition of interest. Thus, the proposed method can be applied to other schemes using LDPC codes for JPEG2000. In particular it can be used together with other schemes such as code rate assignment for unequal error protection (UEP). Comprehensive comparison and extension to other joint designs are left for future research work.

CHAPTER 4

UNEQUAL ERROR PROTECTION AND PROGRESSIVE DECODING

4.1 Introduction

Embedded coding is an attractive feature in many image transmission applications, by which a small subset of the compressed bit-stream can be decoded as an approximation of the original image. It enables progressive transmission and also allows truncation of codestreams without making the correctly received data useless. Examples of popular embedded image coders include Embedded Zerotrees of Wavelets (EZW) [2], Set Partitioning in Hierarchical Trees (SPIHT) [3] and JPEG2000 [4]. However, embedded codestreams are sensitive to transmission errors. It is possible for a single error to cause synchronization loss and even decoding failure in some cases.

In the literature, JSCC has been studied to find optimal error protection schemes for transmission of embedded sources over noisy channels and lossy networks. For example, a progressive image transmission scheme proposed by Sherwood and Zeger [23] employs concatenation of error correction codes and error

detection codes. Another example is a method of protecting SPIHT encoded images transmitted over packet erasure channels proposed by Mohr et al [24], which employs *unequal error protection* (UEP). An efficient UEP mechanism based on Turbo codes as channel codes is proposed by Banister et al. [16]. To find an optimal rate allocation between source and channel codes (i.e., the one minimizing the expected distortion in reconstructed images), the authors use the Viterbi algorithm. In Wu et al. [25], the system of [16] is extended to make use of intrinsic JPEG2000 error resilience tools. Chande and Farvardin [26] provide a dynamic programming solution to the optimization problem of maximizing the average number of source-packets correctly decoded before an error, with $O(R^2)$ time complexity, where R is the transmission rate. A further accelerated algorithm is proposed by Stankovic et al. [27]. A linear-time algorithm is used by Hamzaoui et al. [28] to minimize the expected distortion, by proposing a local search algorithm that starts from a rate-optimal solution and converges to a locally distortion-optimal solution. Lan et al. [29] propose a code design method for irregular repeat accumulate (IRA) codes applied to SPHIT-coded bitstreams. They incorporate the cost function of source distortion into the code design process via density evolution. The resulting system achieves the best published results in terms of average PSNR. However, progressive decoding is not supported, since an entire image is protected by a single codeword, and the whole codeword needs to be received before decoding. A recent survey of

joint source-channel coding techniques can be found in [30].

In this chapter, we propose an unequal error protection scheme for layered JPEG2000 codestreams. The channel codes used in this work are created via the Plotkin construction [31]. This construction builds longer codes from shorter component codes. In the context of unequal error protection, the Plotkin construction has been recently introduced by Kumar and Milenkovic [32]. In the work presented here, we use rate-compatible array codes [33] as component codes. These codes are chosen solely for their simplicity. Stronger codes can be chosen as component codes in the Plotkin construction, which can increase the end-to-end PSNR.

The discussion in this chapter is focused on JPEG2000. However, the proposed techniques are easily adapted to any system that represents images in an embedded fashion. We have chosen JPEG2000 here because of its inherent robustness to bit errors as compared to other embedded coding schemes. Additionally JPEG2000 is being adopted in high profile video and motion picture applications. In particular, JPEG2000 has been chosen by Hollywood studios for future distribution of motion pictures [34]. Additionally, JPEG2000 is being considered for adoption by the WiMedia Alliance. WiMedia UWB is the ultra-wideband platform that will be used for the next generation Bluetooth technology [35].

The main contribution of this work is the proposal of a novel scheduling

strategy based on a multi-stage decoding algorithm. This scheduling strategy leads to progressive decoding of source data, which is especially attractive for embedded sources. As a result, our proposed method provides both efficient unequal error protection and progressive decoding of reconstructed images. In particular, our scheme uses a single long channel codeword to protect an entire image, as in [29]. However our scheme preserves the ability to decode progressively. An inherent feature of embedded source coding is that every newly received packet brings additional information to the source decoder, hence improving the image quality. An additional feature of our scheme is that it enables the channel decoder to exploit new packets to correct residual errors in previously received packets.

In Section 4.2 we give basics of the Plotkin construction. Soft multi-stage decoding of the resulting codes is explained and some modifications are proposed to further improve code performance. In Section 4.3 we introduce novel multi-stage decoding schedules in which the most protected bits are used to help improve the reliability of other bits in belief propagation decoding. Section 4.4 gives examples of constructed codes, and applies the proposed scheme to JPEG2000 image and video sequence transmission. Issues related to the choice of component rate-compatible array codes are also included. Finally, Section 4.5 concludes this work.

4.2 Plotkin Construction of UEP Codes

Recently Dumer [36] introduced a recursive soft decoding algorithm for Reed-Muller codes constructed using the Plotkin construction. His multi-stage algorithm was adapted by Milenkovic et al. [32] and used for decoding UEP codes also constructed by the Plotkin method, but with LDPC codes as component codes. This construction gives a large class of UEP codes supporting a simple multi-stage decoding algorithm. Below, we review the basic UEP code structure from [32], then modify it to further improve the code performance and add more flexibility to the construction.

4.2.1 Review of Construction and Decoding

The Plotkin construction is an algebraic coding theory method to design longer codes from shorter ones [31]. We restrict our discussion to binary linear block codes. A Plotkin-constructed code is a set of codewords each of the form $\mathbf{C} = [\mathbf{u}|\mathbf{u} \oplus \mathbf{v}]$, where $\mathbf{u} \in \mathfrak{C}_1$ and $\mathbf{v} \in \mathfrak{C}_2$ are codewords from two codes, which have the same codeword length n . Codes \mathfrak{C}_1 and \mathfrak{C}_2 are referred to as component codes. The operator \oplus represents the (bit-by-bit) mod-2 sum. The operator $|$ represents concatenation. The resulting codeword thus has length $2n$. If the component codes have parity check matrices H_1 and H_2 , and generator matrices G_1 and G_2 , the code

has parity check matrix and generator matrix of the form:

$$H = \begin{bmatrix} H_1 & 0 \\ H_2 & H_2 \end{bmatrix}, G = \begin{bmatrix} G_1 & G_1 \\ 0 & G_2 \end{bmatrix}.$$

The code can be decoded globally using H . However, the decoding complexity is high because of the large code length. A recursive multi-stage decoding method is introduced in [36] for Reed-Muller codes. In the following description of that decoding algorithm, a memoryless channel with binary input is assumed.

First we introduce some notation. Define the posterior probabilities of the transmitted bit b as $p(b) = \Pr(0|y)$ and $q(b) = \Pr(1|y)$, where y is the received value for that bit. The log-likelihood ratio (LLR) of the bit is calculated as $\lambda(b) = \log(p(b)/q(b))$. The spread h of bit b is defined as $h(b) = p(b) - q(b)$. These quantities are related by

$$p(b) = \frac{e^{\frac{\lambda(b)}{2}}}{e^{\frac{\lambda(b)}{2}} + e^{-\frac{\lambda(b)}{2}}}, \quad q(b) = \frac{e^{-\frac{\lambda(b)}{2}}}{e^{\frac{\lambda(b)}{2}} + e^{-\frac{\lambda(b)}{2}}} \quad (4.1)$$

$$\text{and } h(b) = \tanh\left(\frac{\lambda(b)}{2}\right). \quad (4.2)$$

The source bits are divided into two parts as $\mathbf{m} = [\mathbf{m}^1 | \mathbf{m}^2]$. They are first encoded by component codes individually, $\mathbf{u} = \mathbf{m}^1 G_1 \in \mathfrak{C}_1$, $\mathbf{v} = \mathbf{m}^2 G_2 \in \mathfrak{C}_2$, then combined to form a codeword, $\mathbf{C} = [\mathbf{C}^1 | \mathbf{C}^2] = [\mathbf{u} | \mathbf{u} \oplus \mathbf{v}]$. Thus, if \mathfrak{C}_1 and \mathfrak{C}_2 have rates k_1/n and k_2/n , the code has rate $(k_1 + k_2)/2n$.

The received vector from the output of a communication channel is denoted by $\mathbf{y} = [\mathbf{y}^1 | \mathbf{y}^2]$. This vector is also composed of two parts. The first part \mathbf{y}^1

corresponds to codeword \mathbf{u} , and the second part \mathbf{y}^2 corresponds to $\mathbf{u} \oplus \mathbf{v}$. The LLRs of the bits in \mathbf{u} can be calculated directly using the received vector \mathbf{y}^1 as $\lambda(u_j) = \log(\Pr(0|y_j^1)/\Pr(1|y_j^1))$. However, the LLRs of the bits in component code \mathbf{v} cannot be calculated directly. From [36], the spread of the bit v_j can be calculated as

$$h(v_j) = h(\mathbf{C}_j^1) \cdot h(\mathbf{C}_j^2), \quad (4.3)$$

and the LLR of the bit is

$$\lambda(v_j) = 2 \tanh^{-1}(h(v_j)). \quad (4.4)$$

With these calculated LLRs, \mathbf{v} can be decoded using H_2 .

Note that at this point, there are two versions of the LLR available for u_j . One is calculated from the original received data \mathbf{y}_j^1 , denoted by $\lambda'(u_j)$. The other is denoted by $\lambda''(u_j)$, and is calculated from \mathbf{y}_j^2 in the same way as $\lambda'(u_j)$ is calculated from \mathbf{y}_j^1 , except that the sign is changed whenever the decoded bit $\hat{v}_j = 1$, since $u_j = \mathbf{C}_j^2 \oplus v_j$. Codeword \mathbf{u} can then be decoded using the sum of the two versions of the LLRs.

In summary, the decoding procedure is:

1. Calculate LLRs and spreads for the two parts of the codeword $\lambda(\mathbf{C}^1)$, $\lambda(\mathbf{C}^2)$, $h(\mathbf{C}^1)$, and $h(\mathbf{C}^2)$, using \mathbf{y}^1 and \mathbf{y}^2 , where $\lambda(\mathbf{C}^1)$, $\lambda(\mathbf{C}^2)$ are the LLR vectors

of the bits in the first and the second half of the codeword respectively, and $h(\mathbf{C}^1)$, $h(\mathbf{C}^2)$ are the spread vectors for the bits in the two halves;

2. Calculate LLRs $\lambda(v_j)$ for v using equations 4.3 and 4.4;
3. Decode v using LLRs calculated in step 2, to get \hat{v} ;
4. Calculate two versions of LLRs for u ,

$$\lambda'(u_j) = \lambda(\mathbf{C}_j^1), \quad \lambda''(u_j) = \begin{cases} \lambda(\mathbf{C}_j^2) & \text{if } \hat{v}_j = 0 \\ -\lambda(\mathbf{C}_j^2) & \text{if } \hat{v}_j = 1; \end{cases} \quad (4.5)$$

5. Add both versions of LLRs,

$$\lambda(u_j) = \lambda'(u_j) + \lambda''(u_j); \quad (4.6)$$

6. Decode u using LLRs calculated in step 5, to get \hat{u} .

From an intuitive point of view, unequal error protection arises from the fact that for u , more information is available for its decoder since the codeword is repeated in two places. From an analytical point of view, this can be seen by showing that the effective noise levels experienced by the two component decoders are different. Take BPSK with additive white Gaussian noise as an example. The LLR and spread are then $\lambda(b) = 2y/\sigma^2$, and $h(b) = \tanh(y/\sigma^2)$. For large noise power σ^2 , the first two moments of the spread satisfy the relation [36] $E[h] \sim E[h^2] \sim \sigma^{-2}$. From Equation 4.3, multiplication is involved in calculating the

spread for v , which leads to an equivalent noise power of σ^4 . On the other hand, adding the two versions of the LLR for u leads to an equivalent noise power of $\sigma^2/2$.

The above description is for level-1 construction and decoding. In multi-level construction, each component code is built by sub-component codes in the same fashion as the level-1 construction. The decoding algorithm can be extended to multi-level constructions straightforwardly. For a level- n construction, the codeword is formed iteratively, i.e., each part u and v is constructed by component codes in the same fashion as the code is formed. For example, the level-2 construction would be:

$$C = [u_1 \mid u_1 \oplus v_1 \mid u_1 \oplus u_2 \mid u_1 \oplus v_1 \oplus u_2 \oplus v_2].$$

The decoding follows the same procedure, but the decoded codewords at the end of step 6 are $\hat{u} = [\hat{u}_1 \mid \hat{u}_1 \oplus \hat{v}_1]$ and $\hat{v} = [\hat{u}_2 \mid \hat{u}_2 \oplus \hat{v}_2]$. And the decoding starts again from step 1 to decode the component codes \hat{u}_1 , \hat{v}_1 , \hat{u}_2 and \hat{v}_2 individually.

In Equation 4.6, the first term is from the original received vector y^1 , while the second term is calculated from y^2 and the decoded version of v . Hence, the second term is generally noisier than the first one. It is pointed out in [37] that a threshold function can be used to mitigate this effect. Specifically, if $|\lambda(v_j)|$ is higher than the threshold, the bit v_j is marked as reliable and $\lambda''(u_j)$ is calculated as in Equation 4.5. Otherwise, the bit v_j is marked as unreliable and $\lambda''(u_j) = 0$.

4.2.2 Improved Order of Decoding

In the decoding algorithm described above, the first decoded component (v in level-1 construction) of the code experiences a large amount of noise. In general, this will be true even when the original channel is fairly good. This is because the effective noise level increases along the components of the codeword due to the addition of codewords in the construction (see Equation 4.3). In this work, we modify the decoding algorithm to start from the first part of the codeword, in which case the decoder sees the noise level of the original channel in the decoding of u . Codeword v is then decoded using \hat{u} . After both codewords are decoded, \hat{u} can be further improved. The modified decoding procedure is listed as follows:

1. Calculate LLRs and spreads for the first part of the codeword $\lambda(C^1)$ and $h(C^1)$ using y^1 ;
2. Decode u , to get \hat{u} ;
3. Modify the LLRs $\lambda(C^1)$ according to \hat{u} as

$$\lambda'(u_j) = \begin{cases} |\lambda(C_j^1)| & \text{if } \hat{u}_j = 0 \\ -|\lambda(C_j^1)| & \text{if } \hat{u}_j = 1; \end{cases} \quad (4.7)$$

4. Calculate spreads and LLRs for v from $\lambda'(u)$ and $\lambda(C^2)$ using equations 4.3 and 4.4;
5. Decode v , to get \hat{v} ;

6. Calculate LLRs for u from the decoded \hat{v} ,

$$\lambda''(u_j) = \begin{cases} \lambda(\mathcal{C}_j^2) & \text{if } \hat{v}_j = 0 \\ -\lambda(\mathcal{C}_j^2) & \text{if } \hat{v}_j = 1; \end{cases} \quad (4.8)$$

$$\lambda(u_j) = \lambda'(u_j) + \lambda''(u_j); \quad (4.9)$$

7. Decode u , to get updated \hat{u} .

As before, the algorithm can be extended to multi-level decoding in a straightforward fashion. In this modified algorithm, the best (earliest) part of the codeword is decoded first, and used to help decoding later parts. This improves the results over the original algorithm which starts from the worst (latest) part in the codeword.

4.2.3 Adjustable Component Code Lengths

In the Plotkin construction, all the component codes must have the same codeword length in order to do the bitwise mod-2 sum. In this paper, we provide a method to loosen this restriction in order to make the construction more flexible. This modification broadens the possible choices of component codes. For example, a family of rate-compatible codes may be used as component codes.

Consider level-1 construction as an illustrative example. Suppose component codes \mathcal{C}_1 and \mathcal{C}_2 are of codeword lengths n_1 and n_2 respectively. Without loss of generality, suppose $n_1 < n_2$. Let $u \in \mathcal{C}_1$ and $v \in \mathcal{C}_2$ with $v = [v^1 \mid v^2]$, where partial codeword v^1 has length n_1 , and v^2 has length $n_2 - n_1$. The codeword is

then $\mathbf{C} = [\mathbf{C}^1|\mathbf{C}^2|\mathbf{C}^3] = [\mathbf{u} \mid \mathbf{u} \oplus \mathbf{v}^1 \mid \mathbf{v}^2]$. The received vector also has 3 parts, $\mathbf{y} = [\mathbf{y}^1|\mathbf{y}^2|\mathbf{y}^3]$, having lengths n_1 , n_1 , and $n_2 - n_1$. The decoding algorithm needs to be modified accordingly. In step 4 of the procedure described in the last section, only LLRs for \mathbf{v}^1 are calculated by equations 4.3 and 4.4. The LLRs for \mathbf{v}^2 are calculated from the channel output directly,

$$\lambda(\mathbf{v}_j^2) = \lambda(\mathbf{C}_j^3) = \log \frac{\Pr(0|\mathbf{y}_j^3)}{\Pr(1|\mathbf{y}_j^3)},$$

where $j = 1, 2, \dots, n_2 - n_1$.

4.3 A Multi-Stage Decoding Schedule

As described above, the earlier parts of the codeword are decoded then used to help decode the later parts of the codeword. Subsequently, the later parts can be used to help re-decode the earlier parts. In fact, iterations between the different parts can be carried out for several rounds to improve the decoding performance. We study different decoding schedules for component codes in multi-level constructions. For concreteness, we use a depth-3 construction as an example. The codeword then has the following structure

$$\begin{aligned} \mathbf{C} = & \left[\mathbf{u}_1 \mid \mathbf{u}_1 \oplus \mathbf{v}_1 \mid \mathbf{u}_1 \oplus \mathbf{u}_2 \mid \mathbf{u}_1 \oplus \mathbf{v}_1 \oplus \mathbf{u}_2 \oplus \mathbf{v}_2 \right. \\ & \left. \mathbf{u}_1 \oplus \mathbf{u}_3 \mid \mathbf{u}_1 \oplus \mathbf{v}_1 \oplus \mathbf{u}_3 \oplus \mathbf{v}_3 \mid \mathbf{u}_1 \oplus \mathbf{u}_2 \oplus \mathbf{u}_3 \oplus \mathbf{u}_4 \mid \right. \\ & \left. \mathbf{u}_1 \oplus \mathbf{v}_1 \oplus \mathbf{u}_2 \oplus \mathbf{v}_2 \oplus \mathbf{u}_3 \oplus \mathbf{v}_3 \oplus \mathbf{u}_4 \oplus \mathbf{v}_4 \right]. \end{aligned}$$

Examples of different scheduling could be: $u_1 \rightarrow v_1 \rightarrow u_2 \rightarrow v_2 \rightarrow u_3 \rightarrow v_3 \rightarrow u_4 \rightarrow v_4$; or, $u_1 \rightarrow v_1 \rightarrow u_1 \rightarrow u_2 \rightarrow v_2 \rightarrow u_2 \rightarrow u_3 \rightarrow v_3 \rightarrow u_3 \rightarrow u_4 \rightarrow v_4 \rightarrow u_4$. Many others are possible.

To this end, we define two types of scheduling: one is local scheduling $LS(k)$, and the other is global scheduling $GS(m, n)$. Local scheduling, $LS(k)$, controls the decoder to decode u_k and v_k , for a fixed $k \in \{1, 2, \dots, 2^{(level-1)}\}$. Information is transferred between them back and forth during decoding. First, u_k is decoded, then v_k . The information in v_k can help to refine the previously decoded u_k . Then the refined u_k can further help to decode v_k . Each component can be decoded several times in this iterative fashion. For example, in our application to image transmission in Section 4.4.2, we define a maximum of 7 iterations for u_k and 6 iterations for v_k . It should also be noted that since we use LDPC codes as component codes, each component u_k or v_k is itself decoded iteratively. This iteration is performed using belief propagation, with a maximum of 30 iterations. Global scheduling occurs between different levels, e.g., from level m to level n , denoted by $GS(m, n)$. For example, $GS(1, 2)$ in a scheduling chain of $LS(1) \rightarrow GS(1, 2) \rightarrow LS(2)$, controls the decoder to jump from level 1 (decoding u_1 and v_1), to level 2 (decoding u_2 and v_2).

When deciding which scheduling to use, our criterion is to pass the “cleanest” results along the scheduling chain. For image transmission (as discussed later),

this yields the best reconstruction quality for a given number of received codeword parts. The scheduling used in our applications for a level-3 construction can be represented by the chain:

$$\begin{aligned}
 &LS(1) \rightarrow GS(1, 2) \rightarrow LS(2) \rightarrow GS(2, 1) \rightarrow \\
 &LS(1) \rightarrow GS(1, 2) \rightarrow LS(2) \rightarrow GS(2, 3) \rightarrow \\
 &LS(3) \rightarrow GS(3, 4) \rightarrow LS(4).
 \end{aligned} \tag{4.10}$$

Each node in the chain represents a state in the decoding process. At state $LS(k)$, u_k and v_k are decoded. Index k varies from 1 to $2^{(level-1)}$, which equals 4 here. At state $GS(k, k+1)$, calculations relative to steps 3 – 5 in the decoding procedure are performed. At state $GS(k+1, k)$, calculations relative to steps 6 – 8 are performed. The reason for this choice of scheduling will be more clear in Section 4.4.2.

4.4 Experimental Results

Experimental results are included in this section. Specifically, in Section 4.4.1 we give an example of the Plotkin construction, and demonstrate the UEP property of the resulting code. In Section 4.4.2, we apply the proposed scheme to JPEG2000 image transmission. In Section 4.4.3, results for a JPEG2000 coded video sequence are presented.

4.4.1 UEP

In this section, we show an example of a depth-3 construction. The eight component codes in the code can all be different. By simulation, it was found that three distinct component codes are enough to demonstrate the 8-level UEP property. The v parts in the code experience high noise levels as mentioned before, so strong codes are desired for those parts. In our example, an algebraic geometry code [22] of rate $R_1 = 0.56$ and length 10512 is used for v_k , $k=1, 2, 3$, and 4. A lattice code [22] of the same length is used for u_1 and u_2 . This code has a rate of $R_2 = 0.75$. Another lattice code of the same length is used for u_3 and u_4 , with a rate of $R_3 = 0.78$. The resulting code then has rate $R_P = (R_1 \times 4 + R_2 \times 2 + R_3 \times 2)/8 = 0.66$.

Figure 4.1 shows the BER plot of the constructed UEP code. The eight components are protected unequally, with the BERs gradually increasing from the first component to the last. One can obtain different UEP profiles by choosing component codes of different strength. The decoding scheduling is used as defined in Equation 4.10, and a maximum number of 4 iterations for u_k and 3 iterations for v_k are used at each node $LS(k)$.

Figure 4.2 plots the average BER of the code. The BERs of individual component codes are also plotted (used independently, and not as part of a code). In each case, the channel SNRs are compensated to account for the different code

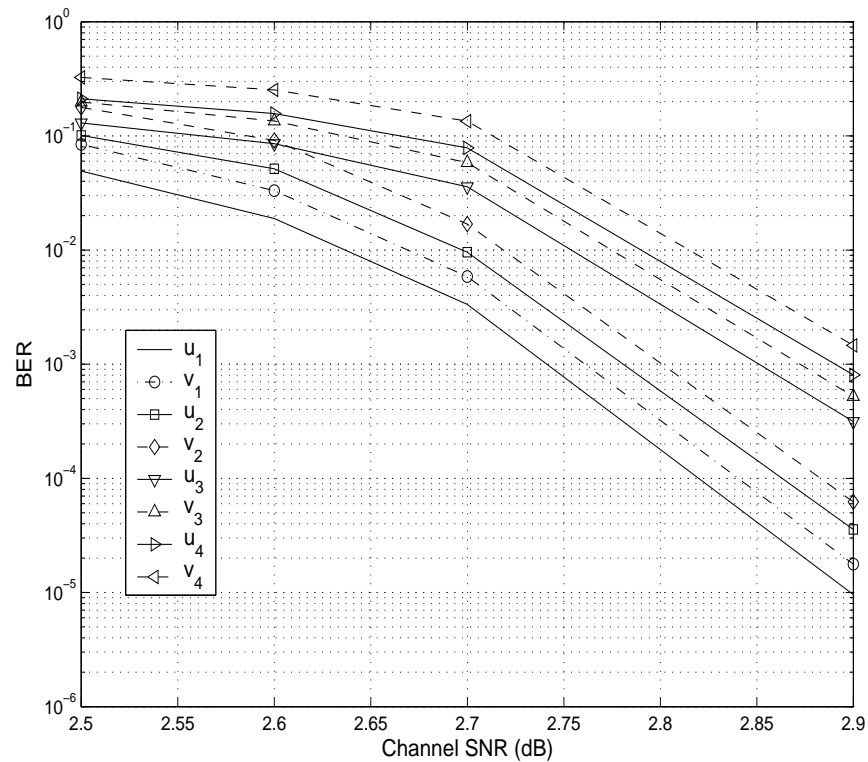


Figure 4.1: BER of different components in binary-input AWGN Channels.

rates. The curve of the UEP code is very steep. Its threshold is controlled by the component codes used in the construction.

4.4.2 Application to JPEG2000 Image Transmission

JPEG2000 provides for progressive transmission using a layered codestream structure. Each layer has a different importance in terms of the contribution it makes to the reconstructed image quality. In this section, we use the UEP scheme offered by the Plotkin construction to protect layered JPEG2000 images. In what

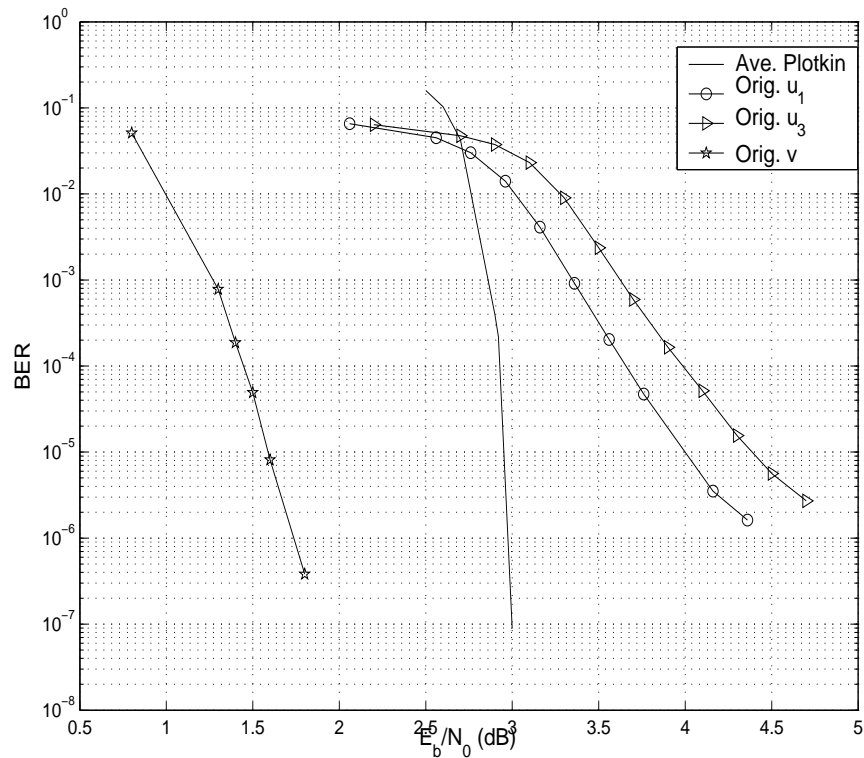


Figure 4.2: BER of UEP code and individual component codes.

follows, an entire image codestream is protected using a single codeword. Specifically, the image is encoded into 8 layers of equal length, and a level-3 code is constructed, in which each part of the codeword protects one layer. A 16-bit CRC outer code is used for error detection for each source layer, with generating polynomial $G(x) = x^{16} + x^{15} + x^2 + 1$. Images are reconstructed using only correctly decoded layers.

The first part of the codeword contains the most important information (layer 0) together with the codestream headers. Each subsequent part protects the next layer, until the last part protects the least important information (layer 7).

The residual error rates after decoding the code are expected to increase from the first layer to the last.

First, we introduce the component codes used in our experiments, and discuss the required optimizations. Next we show the performance of the proposed scheme and compare it to that of equal error protection and the results in [16].

Rate-Compatible Array Codes as Component Codes in the Plotkin Construction

To facilitate our treatment of the joint source/channel optimization problem, we focus on a particular family of codes. In this experiment, we choose our component codes from one family of rate-compatible array codes (RCAC) [33], because they can be described by a small number of code parameters, and have linear encodability. For completeness, a brief introduction to these codes is presented in this section. More detailed construction of rate-compatible array codes can be found in [33] and [38].

Circulants are permutation matrices created by left or right cyclic shifting of the rows of an identity matrix. They are the building blocks for the construction of array codes. Given a $p \times p$ permutation matrix (p prime), the parity check matrix of an array code is specified by two integers j and k indicating the number of building blocks in its rows and columns, with $2 \leq j < k \leq p$. The original

structure of the array codes is given by [39]

$$H_A = \begin{bmatrix} I & I & I & \cdots & I \\ I & \alpha & \alpha^2 & \cdots & \alpha^{k-1} \\ I & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(k-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & \alpha^{j-1} & \alpha^{2(j-1)} & \cdots & \alpha^{(j-1)(k-1)} \end{bmatrix}, \quad (4.11)$$

where α and I are permutation and identity matrices, both of size $p \times p$. In order to achieve efficient encoding, the authors in [33] and [38] modify the structure by cyclically shifting the rows of the matrix H_A in a blockwise manner and setting the lower-triangular elements to zero. The structure is then

$$H = \begin{bmatrix} I & I & I & \cdots & I & I & \cdots & I \\ O & I & \alpha & \cdots & \alpha^{j-2} & \alpha^{j-1} & \cdots & \alpha^{k-2} \\ O & O & I & \cdots & \alpha^{2(j-3)} & \alpha^{2(j-2)} & \cdots & \alpha^{2(k-3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\ O & O & \cdots & I & \alpha^{j-2} & \alpha^{2(j-2)} & \cdots & \alpha^{(j-2)(k-j+1)} \\ O & O & \cdots & O & I & \alpha^{j-1} & \cdots & \alpha^{(j-1)(k-j)} \end{bmatrix}, \quad (4.12)$$

where O is the null matrix of size $p \times p$. The first jp bits of the codeword represent the parity bits, while the other $(k-j)p$ bits represent the information bits.

Rate-compatible array codes are built by puncturing the parity or information bits. A simple way to make a punctured code from the mother matrix H is to puncture parity bits by discarding q rows from the top and q columns from the left. The resulting code has rate

$$R = 1 - \frac{jp - q}{kp - q}.$$

The punctured codes preserve the upper-diagonal structure on the parity bits, thus preserving linear-time encodability. For this reason, only parity-bit-puncturing is

considered in our experiments. Each of the resulting punctured codes have the same number of information bits as the mother code in the family.

Our goal in designing a code is to minimize the average mean squared error (MSE) in a reconstructed JPEG2000 image, given a target total bit rate R_0 under some channel condition. This rate has units of bits-per-pixel and takes into account both the source and parity bits to be transmitted. The optimization is done in two aspects. The first aspect considers different families of RCAC. Optimization is done by choosing different parameters j , k , and p , which decide the rate of the mother code. The second aspect considers puncturing rates within one family, i.e., the optimization decides which punctured codes from the family to be used in the Plotkin construction.

For example, one family of array codes is characterized by parameters $j = 9$, $k = 25$, and $p = 353$. Codes in this family have information length of $K = (25 - 9) \times 353 = 5648$ bits. Now suppose eight component codes are used in a depth-3 Plotkin construction. If the total transmission rate is to be 0.252 bpp for a 512×512 gray level image and 16 bits are used for CRC at each layer, the total length of the constructed code should then be $L = 512 \times 512 \times 0.252 - 16 \times 8 = 65932$ bits. The code rate constructed by the above family is calculated as $R_P = 8K/L = 0.69$.

Suppose the above family is chosen to construct a code of rate 0.69. The

mother code of that family has a code length of $l_M = 25 \times 353 = 8825$. The total number of punctured bits P is determined by the difference between the length that would be obtained if no puncturing were employed and the required code length. In this case $P = 8 \times l_M - L = 4668$ bits. As before, we employ three distinct codes in the construction. The strongest code in the family, which is the mother code, is used for all v parts, and is denoted by $v_k = \{j, k, p, 0\}$, for $k = 1, 2, 3$, and 4. The other two component codes are denoted by $u_1 = u_2 = \{j, k, p, q_1\}$, and $u_3 = u_4 = \{j, k, p, q_2\}$, where q_1 and q_2 are the number of punctured bits in the two codes. Thus the number of punctured bits $P = 2 \times (q_1 + q_2)$. We must then have $q_1 + q_2 = P/2 = 2334$. The values of q_1 and q_2 can then be decided by simulation. The ratio of these two values can be used as a parameter to drive the simulations. The ratio $q_1/q_2 \approx 40/60$ was found to be a good choice. This choice gives strong protection to the earlier layers of the image while yielding good average protection for the whole image. In other simulations, it was found that 8, 9, and 10 are good choices for the j parameter in the code constructions. Too small or too large values of j make the column weights in the punctured codes too low or too high, which will hurt the performance of array codes.

Restricting our attention to RCAC families, the joint optimization is summarized as following:

1. Select a families of RCAC by choosing parameters j , k , and p . The values of j and k decide the rate of the mother code, while the value of p is constrained by the desired length of the code, L .
2. Construct ECC codes using this family. First, the mother code is used for all v components. Next, the values of q_1 and q_2 are decided by simulation for a given channel condition.
3. Repeat the steps above for different families (i.e., different values of j and k).
4. Compare the results between the different families (i.e., different code rates) and choose the one that optimizes the joint source/channel problem.

Performance

Experiments are first performed using binary symmetric channels (BSC). Results are reported for the 512×512 gray level Lenna image for a total transmission rate of 0.252 bpp. For example, for a BSC $\epsilon = 0.03$, the optimal code is constructed from the RCAC family $\{j = 9, k = 25, p = 353, q_1/q_2 = 40/60\}$, denoted as \mathfrak{C}_1 . The code rate is 0.69. Table 4.1 lists the average PSNR results at different states in the scheduling chain. The row indices in the table can be viewed as time indices along the scheduling chain. The PSNR values are listed in eight columns, corresponding to eight layers. For example, the number in the third column represents the quality

Table 4.1: Average PSNR evolution along the scheduling chain for BSC $\epsilon = 0.03$ at 0.252 bpp, EF – error free case.

Time	u_1	v_1	u_2	v_2	u_3	v_3	u_4	v_4
1	18.6	19.2	-	-	-	-	-	-
2	23.2	26.4	-	-	-	-	-	-
3			27.4	28.1	-	-	-	-
4			27.9	29.1	-	-	-	-
5	23.2	26.4			-	-	-	-
6			27.9	29.1	-	-	-	-
7					29.2	29.2	-	-
8					30.0	30.9	-	-
9							31.5	32.1
10							31.5	32.1
EF	23.2	26.4	27.9	29.2	30.1	31.0	31.7	32.3

of the images, which are reconstructed by the first three layers.

The first two rows correspond to the first state in the scheduling chain, $LS(1)$, and list the image quality obtained by decoding the first two layers. The improvements obtained from the first row to the second row illustrate the novel progressive decoding ability of the proposed scheme. After receiving the first layer, LDPC decoding is performed for the first component code u_1 . The image reconstructed by the first layer has a PSNR value of 18.6 dB. Then the second layer is received, and v_1 is decoded. The image constructed by the first two layers has a better PSNR of 19.2 dB. The iterative refining procedure between u_1 and v_1 is then performed, which yields much better image quality at the end of state $LS(1)$.

Specifically, the PSNR values are increased to 23.2 and 26.4 dB, as shown in the second row. As mentioned before, the scheduling tries to get the “cleanest” possible results for earlier components before it decodes the later ones. The second row gives approximately the best quality available, given that only the first two layers are received.

After the next two layers are received, the decoding goes on to the second and the third states of the scheduling chain, i.e., $GS(1, 2)$ followed by $LS(2)$. Similarly, at state $LS(2)$, the qualities corresponding to layers 3 and 4 are refined from row 3 to row 4. After $GS(2, 1)$, rows 5 and 6 record the results at states $LS(1)$ and $LS(2)$ respectively. These steps are intended to further clean up the residual errors in the first four layers as much as possible. Similarly, rows 7 and 8 correspond to $LS(3)$, and rows 9 and 10 correspond to $LS(4)$. The last row in Table 4.1 lists the PSNRs of the error-free layers.

For the purpose of comparison, we use individual codes to protect each source layer. This strategy offers progression at the source coding level, but is not capable of offering progression at the channel coding level. An EEP scheme using this strategy is tested. To this end, a reference code is selected from the same family used for Plotkin construction, with code rate the same as the constructed code. This code is used to individually protect each layer in the image.

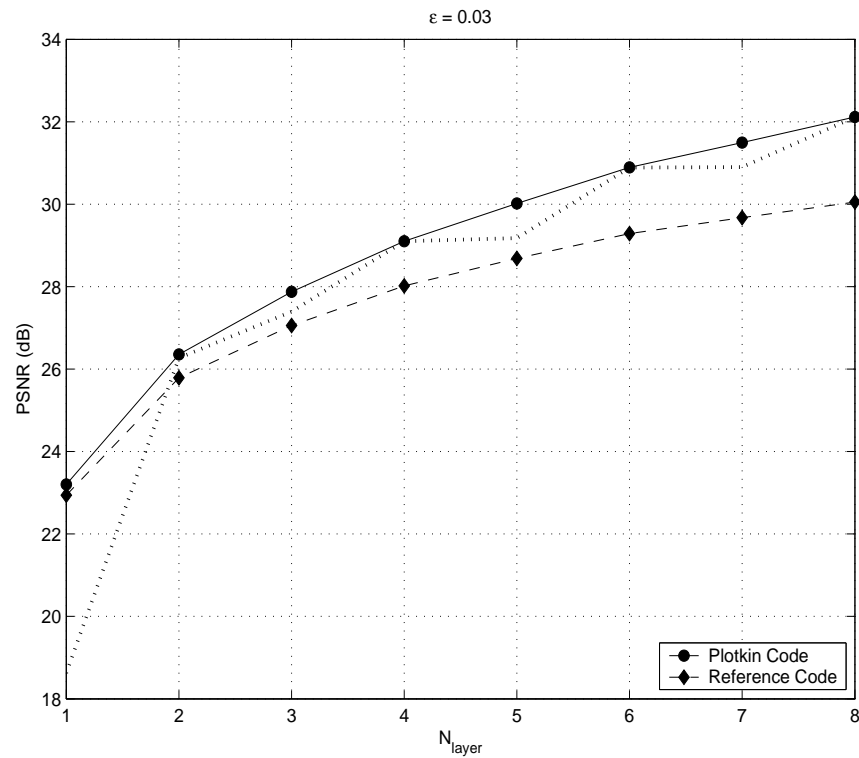


Figure 4.3: Average PSNR vs. number of received packets for BSC $\epsilon = 0.03$ at 0.252 bpp.

The comparison is shown in Figure 4.3. The horizontal axis represents the number of layers used to reconstruct the image. The dotted line shows the best results (from Table 4.1) obtained when only the indicated number of layers have been received. The solid line shows the best performance available when all layers are available. The dashed curve for the reference code in the figure corresponds to the result of EEP. It is clear that the proposed UEP scheme gives better protection than the EEP scheme, even though the average BER of the EEP code is better than the average BER of the UEP code.

Table 4.2: Average PSNR from UEP and EEP for BSC channels at 0.252 bpp.

	$\epsilon = 0.03$	$\epsilon = 0.01$
UEP	32.12	33.00
EEP	30.05	31.57
[16]	31.90	32.56

BSC with $\epsilon = 0.01$ is also tested. The optimal code \mathfrak{C}_2 has parameters $\{j = 10, k = 50, p = 173, q_1/q_2 = 40/60\}$. The code rate is 0.84. Table 4.2 lists the ultimate PSNR results of the proposed progressive decoding scheme when all data are decoded. The EEP results are reported as well. Also included in the table are the results from [16], at the same transmission rate and channel conditions.

4.4.3 Application to Video Transmission

Finally we apply the proposed scheme to video sequence transmission. Part III of the JPEG2000 standard (referred to as Motion JPEG2000) is intended for image sequences. Each frame is encoded independently in Motion JPEG2000, which provides a high degree of “interframe error robustness” during transmission. The techniques of this dissertation provide for “intraframe robustness.” In our experiments, the NTSC-based SIF format “football” sequence is chosen as the test sequence. The sequence contains 125 frames, each of size 352×240 . Only the luminance component of the sequence is used here. Each frame is encoded into 4 layers. A Plotkin construction with 4 protection levels is used to protect each frame. The Plotkin

Table 4.3: Average PSNR evolution along the scheduling chain for AWGN=2.8 dB, EEP – Equal error protection.

Time	u_1	v_1	u_2	v_2
1	15.8	15.9	-	-
2	19.6	21.8	-	-
3			21.8	21.8
4			23.1	24.0
5	19.6	21.9		
6			23.1	24.1
EEP	19.5	21.6	22.7	23.5

construction has parameters $\{j = 8, k = 19, p = 487, q_1/q_2 = 44/56\}$. The parameters are optimized for an AWGN channel of SNR=2.8 dB. The total length of the code is $L = 33268$, which yields a total transmission rate of $(L + 4 \times 16)$ bits/frame $\times 30$ frames/second =1M bits/second. The same scheduling chain is used to control the multi-stage decoding as before, except that only four component codes u_1, v_1, u_2 and v_2 are included. The PSNR results are listed in Table 4.3. The last line in the table lists the results from the EEP scheme. The PSNR values are calculated from averaged MSE from all frames.

4.5 Conclusions

In this chapter, we present a UEP scheme based on the Plotkin construction. The proposed scheme offers two levels of progressive decoding: the first is the source

coding level, the second is the channel coding level. At the source coding level, more and more source data are available to improve the reconstructed image quality. At the channel coding level, later received data carry parity information to help reduce the BER of earlier received data. The experiments show encouraging results. By adjusting the scheduling and iteration times, a trade-off can be achieved between the code performance and the decoding complexity. Although the end-to-end PSNR in our proposed scheme is less than that reported in [29] (0.62 dB at 0.252 bpp for BSC $\epsilon = 0.03$), our scheme has the advantage of progressive decoding. In this chapter, we have focused on demonstrating this novel ability, while choosing rate-compatible array codes as component codes solely for the sake of simplicity. Other stronger codes can be chosen as the component codes in the Plotkin construction, which can increase the end-to-end PSNR of our system.

CHAPTER 5

JOINT RATE ALLOCATION IN MULTICHANNEL SYSTEMS

5.1 Introduction

Parallel channels are often used in multimedia transmission for more reliable communication or more efficient bandwidth usage. For example, in wireless channels, fading, shadowing or other sources of interference can cause fluctuation of channel quality. Data transmitted through such a channel may have large distortion, or even be completely useless, especially for error-sensitive sources such as compressed images. Another example is an underwater acoustic channel. In such a channel, there exist frequency-dependent signal attenuation and multipath due to reflections from the surface and the bottom of the sea. Intersymbol interference (ISI) results from the nonideal channel frequency response, and a complex equalizer may be needed to compensate channel distortion. Using multiple parallel channels is one approach to combat such problems. It has attracted significant research interest, and has yielded many modern communications techniques, such as multicarrier modulation (MCM) or discrete multitone (DMT) [40], [41]. These techniques are well known for their advantages of bandwidth efficiency, excellent ISI performance, and immunity

to multi-path fading or time dispersion.

In multi-channel systems, subchannels can have differing characteristics, such as differing signal-to-noise ratios (SNR) or error rates. In the literature, allocation algorithms allocate power and bit rate (e.g., the constellation size) such that the symbol error rate (SER) or bit error rate (BER) is the same among subchannels. In joint designs for multimedia transmission, many algorithms aim to optimize the system performance for a given *average* SNR over subchannels, entirely ignoring specific subchannel variations. In some cases, varying subchannel SNR is considered, but not fully utilized for a joint design. In this chapter, the potential of varying SNR over all subchannels is exploited to design a joint source-channel coding scheme for robust transmission of scalable sources in multi-channel systems. A brief literature review on allocation algorithms in multi-channel systems will be given in the following paragraphs.

It is well known that the “water-filling” energy distribution is capacity-achieving [8], under the assumption of infinite granularity of the constellation size. Hughes-Hartogs [42] proposes an alternative finite granularity loading method. But the algorithm is very slow when a large number of bits need to be allocated, such as in asymmetric digital subscriber line (ADSL) applications. A practical algorithm is proposed by Chow et al. [43]. It is iterative and is a rounded approximation of the “water-filling” method. The optimization criterion is based on the channel capacity

of the subchannels. Fischer and Huber [44] use another optimization criterion that maximizes the signal-to-noise ratio (minimizes the error rate) in each subcarrier. This criterion is better because in the capacity-achieving method, the bit rate (constellation size) and signal power are directly related, which results in less room for optimization. Other optimization criteria are used in the literature. Krongold et al. [45] allocate power efficiently to maximize data rate and system performance margin. The proposed fast algorithm uses lookup tables and Lagrange multiplier bisection searches for a given minimum symbol error rate requirement.

As mentioned in previous chapters, JSCC can improve system performance for multimedia transmission. Modern error correction codes (ECC) are widely used along with MCM modulation techniques to lower error rates in communication systems. Unequal error protection (UEP) schemes are also desirable for multimedia transmission, because advanced source coding techniques often yield source bits having different importance which inherently appeal to UEP. Pradhan and Ramchandran [46] propose a multi-resolution modulation to achieve UEP for different resolution layers, but within each layer, equal error protection (EEP) is assigned across subchannels. Turbo-coded OFDM-based interactive video telephony is proposed by Cherriman et al. [47]. A modulation mode is chosen to maximize throughput while keeping estimated BER lower than a target BER. In addition to the frequency diversity provided by MCM modulation, other diversities, such as space

and time diversity, are also studied and utilized in communication systems. For example, space-time coding is a popular technique used to achieve space diversity in wireless communication systems. An OFDM system with spatial diversity is proposed for image transmission by Song and Liu [48]. Their scheme uses space-time block codes (STBC) and Reed Solomon codes. Parallel subchannels are assumed to have the same SNR. Source and channel rates are jointly optimized to maximize the expected number of source bits correctly received. Sun et al. [49], [50] propose the transmission of scalable images over space-time coded OFDM systems. Fast joint source-channel rate allocation is performed for a given average SNR (BER). Space-time coded OFDM systems for video transmission over wideband wireless channels are also proposed in [51]. Chan et al. [52] propose to combine multiple description coding and frequency diversity. Parity levels in the FEC-based multiple description coding are optimized for a given SNR.

The above algorithms can be categorized as aiming to optimize system performance for a given *average* SNR (over subchannels). Varying subchannel SNR is considered in some cases, but not fully utilized for a joint design. Transmission of a layered source over spectrally shaped channels using MCM is proposed by Zheng and Liu [53]. The source layers are predetermined before allocation. Subchannels are ordered by SNR and the earlier layers select subchannels of higher SNR. The number of subchannels used by each layer is governed by the length of that layer.

Zhao et al. [54, 55] propose optimal rate and power allocations in multichannel systems. Subchannels are assigned to layered images/videos sequentially first, then rates and power levels are allocated jointly between channels.

Rate-based optimization for the single channel case has been proposed by Stankovic et al. [56]. In this work, we tackle the problem of designing a rate-based optimization scheme for multiple parallel channels, where the potential of varying SNR over all subchannels is fully exploited for robust transmission of scalable sources via fixed-length packets. For each source packet, the proposed algorithm selects a subchannel among all available subchannels, and chooses a channel code rate, such that the expected length of the correctly received source data is maximized. Additionally, our algorithm can incorporate constraints on the number of packets that can be carried by any given channel. Such constraints can not be handled by the single channel solution. Additionally, we investigate transmitting multiple sources. Source multiplexing is introduced and combined with rate allocation to provide a more fair distribution of quality among transmitted sources in addition to minimizing the average MSE over all sources.

We verify the proposed algorithm for different channel models. Two models are chosen for this purpose. The first one is a model of independent parallel Gaussian channels. This model is chosen because it can represent a non-white Gaussian noise channel or other spectrally shaped channels arising in various applications, such as

DMT or MCM, multiple input multiple output (MIMO) systems, and OFDM, etc. OFDM is a special case of MCM. Coded OFDM (COFDM) has been specified for digital audio broadcasting (DAB) [57] and digital video television (DVB-T) [58]. The second model is chosen to demonstrate the suitability of the proposed scheme in a specific channel. To this end, a plastic optical fiber (POF) model is selected. POF is a newly developed type of optical fiber. One major advantage of POF over glass fiber is that it is inexpensive and simple to install. POF can be used for Gigabit Ethernet and local area networks (LAN) [59], [60]. It can also be used in high definition television (HDTV) and internet protocol television (IPTV) delivery systems [61]. Randel et al. [62] provide results with 1 Gbit/s transmission in a step-index POF link using adaptive multiple subcarrier modulation. In our work, we apply optimal rate allocation in the POF channel with OFDM modulation.

The optimization problem is formulated and solved in Section 5.2. In Section 5.3, the proposed algorithm is applied to image transmission. The two channel models mentioned above are tested. Section 5.4 concludes the chapter.

5.2 Optimization Problem

Suppose that scalable sources are transmitted via fixed-length packets. There are a total of N packets transmitted over M subchannels. Subchannel m transmits N_m channel packets, with $\sum_{m=1}^M N_m = N$. Often $N_m = N/M$, but we treat the more

general case here, where N_m can be different for different subchannels. Define the set of available channel code rates as $\mathbf{R} = \{r_1, r_2, \dots, r_L\}$ and the set of subchannels as $\mathbf{S} = \{s_1, s_2, \dots, s_M\}$. A code rate from \mathbf{R} and a subchannel from \mathbf{S} are chosen for each packet. This allocation is denoted by a rate assignment scheme $R = ((r_{k_1}, s_{k_1}), (r_{k_2}, s_{k_2}), \dots, (r_{k_N}, s_{k_N}))$. Let $E[\cdot]$ denote the expectation operator, and $\Delta D(R)$ be the distortion reduction associated with a rate assignment scheme R . $\Delta D_i(R)$ represents the distortion reduction associated with the event that the first error happens in the $(i + 1)^{th}$ packet under rate assignment scheme R , and $P_i(R)$ represents the probability of this event. The expected distortion reduction for the reconstructed sources at the receiver can then be written as

$$E[\Delta D(R)] = \sum_{i=1}^N P_i(R) \Delta D_i(R), \quad (5.1)$$

where

$$P_i(R) = p(r_{k_{i+1}}, s_{k_{i+1}}) \prod_{j=1}^i (1 - p(r_{k_j}, s_{k_j})), \quad (5.2)$$

and

$$\Delta D_i(R) = \sum_{j=1}^i \Delta d_{k_j}. \quad (5.3)$$

In these equations, $p(r_{k_j}, s_{k_j})$ is the packet error probability of the j th packet, with a channel code rate of r_{k_j} in subchannel k_j ; and Δd_{k_j} is the distortion reduction provided by the source bytes in the j th channel packet, when correctly received. This expression assumes progressive decoding, and that if one or more errors are detected in one channel packet, this packet and all following packets are discarded.

The distortion optimization problem of (5.1) is difficult to solve in the single channel case, and is even more difficult for the multi-channel case, since the search space is expanded from N^L to $N^{L \times M}$. The number M is usually large, e.g., 128 for a practical OFDM system. In such cases, the expansion of the search space from single channel to multi-channel is prohibitively complex.

In this chapter, we consider an alternative rate-based optimization problem. It has been shown by Hamzaoui et al. [63] for the single channel case, that the rate-optimal solution provides a good approximation to the distortion-optimal solution for efficient embedded coders. Chande and Farvardin [64] proposed a rate-optimal solution for fixed-length information blocks (variable length packets). Stankovic et al. [56] extended it to fixed-length packets and proposed a fast algorithm. In this chapter, we study the multi-channel optimization problem, and derive a linear time rate-optimal solution. Fixed-length channel packets are assumed. Such packets are suitable in many practical applications, such as in asynchronous transfer mode (ATM) networks.

The problem can be reformulated as follows. Let $E_N(R)$ denote the expected value of the correctly received source length associated with rate assignment scheme R . $L_i(R)$ represents the received source length associated with the event that the first error happens in the $(i + 1)^{th}$ packet under rate assignment scheme R , and $P_i(R)$ represents the probability of this event. A rate-optimal error protection

scheme (EPS) then maximizes the expected length of correctly received data,

$$E_N(R) = \sum_{i=1}^N P_i(R) L_i(R), \quad (5.4)$$

where

$$L_i(R) = \sum_{j=1}^i l(r_{k_j}), \quad (5.5)$$

and $l(r_{k_j})$ is the number of source bytes in one packet, which is determined by the channel code rate assigned to the packet.

We give two lemmas and two corollaries before we derive the optimal solution.

Lemma 1: Let $R = ((r_{k_1}, s_{k_1}), (r_{k_2}, s_{k_2}), \dots, (r_{k_N}, s_{k_N})) \in (\mathbf{R} \times \mathbf{S})^N$ be an N -packet EPS. Then

$$\begin{aligned} & E_N[(r_{k_1}, s_{k_1}), \dots, (r_{k_N}, s_{k_N})] \\ &= E_1[(r_{k_1}, s_{k_1})] + (1 - p(r_{k_1}, s_{k_1})) \\ & \quad \times E_{N-1}[(r_{k_2}, s_{k_2}), \dots, (r_{k_N}, s_{k_N})], \end{aligned} \quad (5.6)$$

where $E_1[(r_{k_1}, s_{k_1})] = l(r_{k_1})(1 - p(r_{k_1}, s_{k_1}))$.

Proof: The proof is given in the Appendix.

Lemma 2: If the $(N-1)$ -packet EPS $((r_2^*, s_2^*), \dots, (r_N^*, s_N^*))$ is rate optimal, and if $E_N[(r_1^*, s_1^*), (r_2^*, s_2^*), \dots, (r_N^*, s_N^*)] \geq E_N[(r_k, s_k), (r_2^*, s_2^*), \dots, (r_N^*, s_N^*)]$ for all $r_k \in \mathbf{R}$, and $s_k \in \mathbf{S}$, then the N -packet EPS $((r_1^*, s_1^*), (r_{k_2}, s_{k_2}), \dots, (r_N^*, s_N^*))$ is rate optimal.

Proof: The proof is given in the Appendix.

The next two corollaries follow from these lemmas. If the N -packet EPS

$((r_1^*, s_1^*),$

$\dots, (r_N^*, s_N^*))$ is rate optimal, then we have the following corollaries.

Corollary 1: For $1 \leq i \leq N - 1$, the $(N - i)$ -packet EPS $((r_{i+1}^*, s_{i+1}^*), \dots, (r_N^*, s_N^*))$

is rate optimal.

Proof: The proof is given in the Appendix.

This result shows that an N -packet EPS can be obtained by first calculating 1-packet EPS (which is trivial), and recursively adding one packet ($N-1$) times.

Corollary 2: Denote subchannel s_n having a smaller bit error rate than s_m by $s_n > s_m$. Then $s_1^* \geq s_2^* \geq \dots \geq s_N^*$.

Proof is given in the Appendix.

From the above two corollaries, the rate-based optimization algorithm can be formulated as follows:

Algorithm:

1. Order the M subchannels, so that $SC_1 \geq \dots \geq SC_m \geq \dots \geq SC_M$. The subchannel SC_m contains N_m packets, and $\sum_{m=1}^M N_m = N$.
2. Set $i = 1$, $b = M$, $s_{N-i+1} = SC_b$, and $j_{N-i+1} = \arg \max_{k=1, \dots, L} E_1[(r_k, SC_M)]$.

3. $i = i + 1$;

If $i = N_b + N_{b+1} + \dots + N_M + 1$, then $b = b - 1$, $s_{N-i+1} = SC_b$;

If $i = N + 1$, then $\{(r_1^*, s_1^*), (r_2^*, s_2^*), \dots, (r_N^*, s_N^*)\} = \{(r_{j_1}, s_1), (r_{j_2}, s_2), \dots, (r_{j_N}, s_N)\}$, and stop.

4. $j_{N-i+1} = \arg \max_{k=1, \dots, L} E_i[(r_k, s_{N-i+1})$,

$(r_{j_{N-i+2}}, s_{N-i+2}), \dots, (r_{j_N}, s_N)]$; go to step 3.

In the above algorithm, the assignment starts from the worst subchannel in step 2 (from Corollary 2). The N -packet EPS is derived from a 1-packet EPS recursively (from Corollary 1). When each of the packets in the current subchannel has been assigned a channel code rate, the packets in the next worst subchannel will be assigned. The procedure continues until all N packets in all M subchannels are assigned code rates.

5.3 Applications to Image Transmission

Now that we have derived a fast rate-optimal rate allocation algorithm, in this section, we apply the proposed algorithm to image transmission. Two channel models are used as examples. The first assumes independent parallel Gaussian channels. The second derives from a plastic optical fiber (POF). The first channel model is

chosen as a general example, which can represent various environments such as non-white Gaussian noise channels or spectrally shaped channels. The second channel model is chosen because of its increasing popularity in optical communication systems. Before discussing the two cases individually in Sections 5.3.3 and 5.3.4, we describe the image coding and channel coding techniques used in the experiments.

5.3.1 Source and Channel Coding

JPEG2000 coded images are used as source images. The optional JPEG2000 packed packet header marker segments (PPM) are used in our experiments to move all JPEG2000 packet headers to the main header. The main header (including the relocated packet headers) is then assigned the lowest code rate (highest protection).

In what follows, we consider the case where multiple images are transmitted through a common channel. These images may be entirely unrelated. However, in Part III of the JPEG2000 standard (referred to as Motion JPEG2000), each frame of a video sequence is compressed independently. Thus, the multiple images transmitted through the channel (using our proposed system) could be frames of a video.

We examine two methods to transmit multiple images. We first consider transmitting each image with an equal number of channel packets. For example, if 256 packets are used to transmit 4 images, each image occupies 64 packets. In this

case the total available resources are divided equally. We next consider multiplexing the images together and adjusting the available resources between the images according to their characteristics. The multiplexing is done by combining all the layers from all the images. All the main headers are also combined and placed at the beginning of the resulting codestream. Since each layer is associated with a threshold slope value, all the layers (from all images) can be ordered by their threshold slope values. The combined codestream can be viewed as a large “pseudo” image.

Packetization and rate allocation can then be performed on the “pseudo” image using the algorithm described in Section 5.2 above. At the channel output, the “pseudo” image is split into individual images, each of which is decoded separately by a JPEG2000 decoder. If one or more errors are detected in one channel packet, this packet and all the following packets are discarded. This procedure is based on the assumption that later layers cannot be decoded without the reception of earlier layers. This assumption is not strictly satisfied since an error in an image will not affect the decoding of another image. Nevertheless, we make the assumption here for the sake of simplicity.

5.3.2 Turbo Codes

Rate-compatible punctured turbo codes (RCPT) [65] are employed as channel codes. This section gives a brief introduction on Turbo codes. Turbo codes were first

presented by Berrou et al. [5] in 1993. Their near Shannon limit error correction capability has made them an active research topic.

A Turbo code encoder consists of a concatenation of two (or more) convolution encoders. Recursive systematic convolutional (RSC) codes are common choices of the constituent codes. A parallel concatenated Turbo encoder and decoder structure is shown in Figure 5.1. Two encoders are separated by a pseudo-random permuter or interleaver. Information bits are sent to the channel, together with parity bits. An optional puncturing mechanism can be used for parity bits; thus rate compatible Turbo codes (RCTC) can be constructed.

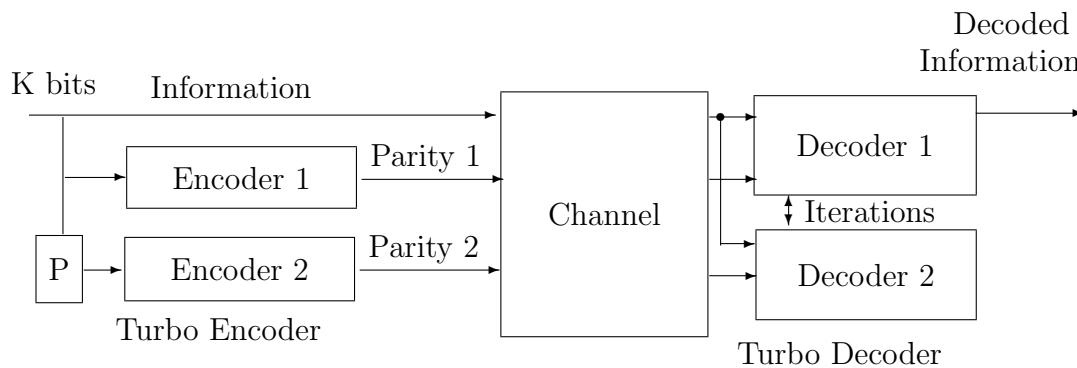


Figure 5.1: A Turbo encoder and decoder diagram.

The existence of permutation forbids the use of Viterbi algorithm as used in decoding convolutional codes, for the structure of a turbo code trellis would be too complicated. However, a suboptimum iterative decoding algorithm offers

near maximum-likelihood (ML) performance. As in Figure 5.1, two soft-in/soft-out (SISO) [65] decoders communicate probabilistic information iteratively. The goal of the decoder is to estimate the *a posteriori* probabilities (APP), $\Pr(u_k|y)$, $k = 1, 2, \dots, K$. Vector y is the received noisy codeword. Suppose u_k takes a value in the set $\{+1, -1\}$. Optimal decisions can be made by the maximum *a posteriori* (MAP) criterion

$$\frac{\Pr(u_k = +1|y)}{\Pr(u_k = -1|y)} \stackrel{+1}{\underset{-1}{\gtrless}} 1.$$

To avoid numerically unstable computation, “log domain” calculations are usually employed. In that case the log-likelihood ratio (LLR) is calculated, which is defined as

$$L(u_k) \triangleq \log \left(\frac{\Pr(u_k = +1|y)}{\Pr(u_k = -1|y)} \right).$$

The BCJR [66] algorithm is used to compute the LLR $L(u_k)$. The algorithm computes the LLR of each symbol through forward/backward recursions on the code trellis. The constituent decoders then exchange extrinsic information iteratively.

A rate 1/3 parallel concatenated convolutional code with generator polynomial (33,31) is chosen as the encoder. Puncturing patterns are chosen from [67] with a puncturing period of eight. A maximum number of 20 iterations are used for decoding. Channel packets of length 512 bytes are used. Four bytes are reserved for 32-bit CRC, which is used for error detection following turbo decoding. One byte is reserved for transmitting side information such as the turbo code rate. Code rates

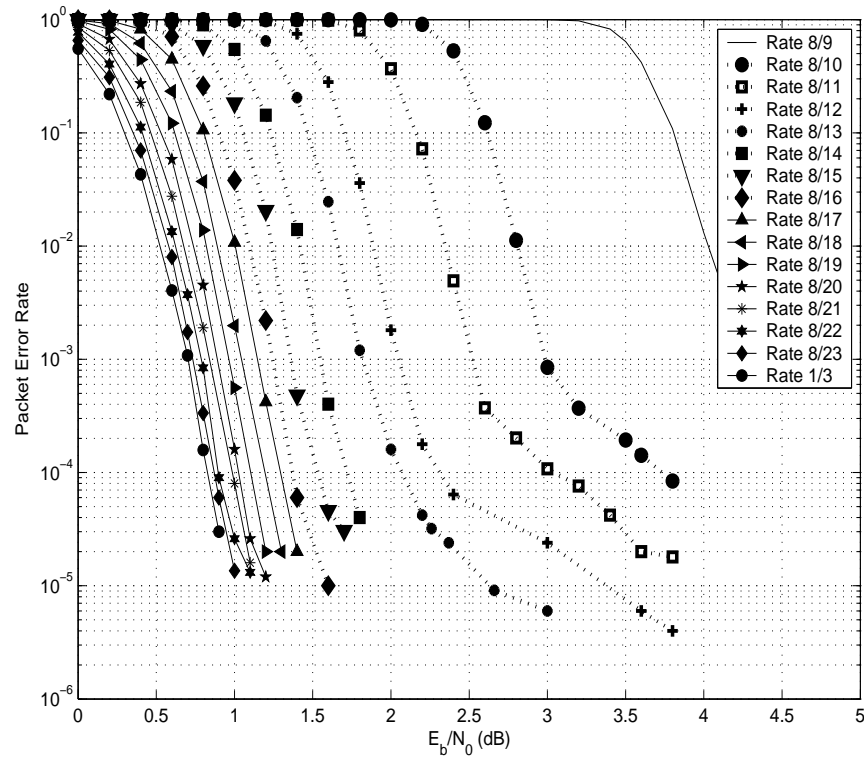


Figure 5.2: Packet error rates in AWGN channel.

are chosen from the set $R = \{8/9, 8/10, \dots, 8/23, 8/24\}$. The cardinality of this set is 16. The following subsections will discuss transmission for two different channel models.

5.3.3 Independent Parallel Gaussian Channels

As a first example, independent parallel Gaussian channels are assumed. BPSK modulation is used. Packet error rates for the turbo codes in an AWGN channel are shown for different channel conditions in Figure 5.2. Four greylevel 512×512 images are used as source images: Lena, Goldhill, Whitehouse, and Lighthouse. We

choose 32 AWGN channels with equally spaced SNRs between 1.0 dB and 3.0 dB.

There are five error protection schemes tested in the experiments. The first is a UEP scheme. Multiple source images are multiplexed together using the method described in Section 5.3.1. Then the rate allocation algorithm described in Section 5.2 is used to assign a channel and code rate to each packet. This scheme is denoted as Scheme I henceforth. The second scheme is an EEP scheme, denoted by Scheme II. In Scheme II, multiple source images are multiplexed as in Scheme I. However, data from the “pseudo” image is packed into packets, each of which is protected by the same code, with a code rate equal to the average code rate used in Scheme I. This scheme is denoted as Scheme II. Next we consider a UEP scheme, in which images are not multiplexed together. This scheme is denoted as Scheme III. In this scheme, the images are not multiplexed. Rather, each source image is assigned the same number of channel packets from each subchannel. Before transmission of each image, the rate allocation algorithm introduced in Section 5.2 is first used to assign subchannels and code rates to all the packets of that image. In Scheme IV, images are not multiplexed, and all packets are protected by the same code, with a code rate equal to the average code rate used in Scheme III. Schemes III and IV can be thought of as time division schemes. That is, the first image can be thought of as being transmitted using all channels, followed by the second image, and so on. Scheme III optimizes the use of the channels (UEP) while Scheme IV does not

(EEP). Lastly, Scheme V is tested. In this scheme, images are multiplexed as in Scheme I. But instead of using the true SNRs of all the subchannels, an average channel SNR is used in the rate allocation algorithm. In this scheme, a 2.0 dB channel is assumed for allocating all N packets. In all the schemes, headers are protected by a code of rate $1/3$.

Simulation results are listed in Table 5.1. A total number of $N = 128$ packets are used for an average rate of 0.5 bpp/source, including all header and parity info. Similarly, $N = 256$ packets are used for an average rate of 1.0 bpp/source. Each case is simulated 1000 times. The resulting MSE values for each case are averaged and then converted to PSNR values. Similarly, the row labelled “Average” is MSE averaged over all simulations and all images and then converted to PSNR. This method has been adopted in several recent studies [50, 68, 25], and has been shown to correlate well with visual quality [69]. UEP gains can be observed by comparing results from Scheme I with Scheme II (about 0.8 to 1 dB), and Scheme III with Scheme IV (about 0.5 dB). Source multiplexing gains can be observed by comparing results from Scheme I with Scheme III, and Scheme II with Scheme IV. These gains come from differing numbers of packets being allocated between different sources according to their characteristics. The total gain from both multiplexing and UEP is about 1.2 dB at 0.5 bpp/source, and 2.0 dB at 1.0 bpp/source. It can also be observed that the proposed multiplexing tends to give a more fair

Table 5.1: Average MSE and PSNR for parallel Gaussian channels.

0.5 bpp/source										
	Scheme I		Scheme II		Scheme III		Scheme IV		Scheme V	
	MSE	PSNR	MSE	PSNR	MSE	PSNR	MSE	PSNR	MSE	PSNR
Lena	68.66	29.76	86.74	28.75	26.41	33.91	30.06	33.35	90.61	28.56
Goldhill	102.01	28.04	133.53	26.88	54.59	30.76	63.26	30.12	135.73	26.80
Whitehouse	289.71	23.51	354.77	22.63	355.67	22.62	419.24	21.91	367.72	22.48
Lighthouse	470.34	21.41	545.16	20.77	655.16	19.97	716.13	19.58	616.18	20.24
Average	232.68	24.46	280.05	23.66	272.96	23.77	307.17	23.26	302.56	23.33
1.0 bpp/source										
	Scheme I		Scheme II		Scheme III		Scheme IV		Scheme V	
	MSE	PSNR	MSE	PSNR	MSE	PSNR	MSE	PSNR	MSE	PSNR
Lena	48.64	31.26	55.86	30.66	11.87	37.39	13.72	36.76	56.28	30.63
Goldhill	74.89	29.39	89.32	28.62	28.87	33.53	34.30	32.78	91.68	28.51
Whitehouse	186.37	25.43	214.55	24.82	206.95	24.97	238.44	24.36	226.34	24.58
Lighthouse	177.61	25.64	253.50	24.09	440.83	21.69	489.96	21.23	309.07	23.23
Average	121.88	27.27	153.31	26.28	172.13	25.77	194.11	25.25	170.84	25.80

QoS among multiple sources in the sense that the MSE differences are significantly reduced. Comparing Schemes I and V, it can be observed that significant gains (about 1.1 dB for 0.5 bpp/source and 1.5 dB for 1.0 bpp/source) are achieved by fully utilizing the individual characteristics of all subchannels, rather than simply designing for the “average” channel.

5.3.4 Plastic Optical Fiber Channel

In addition to the general model of independent parallel Gaussian channels discussed in the previous section, a graded-index plastic optical fiber (GI-POF) channel model with OFDM modulation is considered as the second example.

The impulse response of GI-POF can be modelled as [70]

$$h(t) = \frac{1}{\sqrt{2\pi\alpha^2}} \exp\left(-\frac{t^2}{2\alpha^2}\right). \quad (5.7)$$

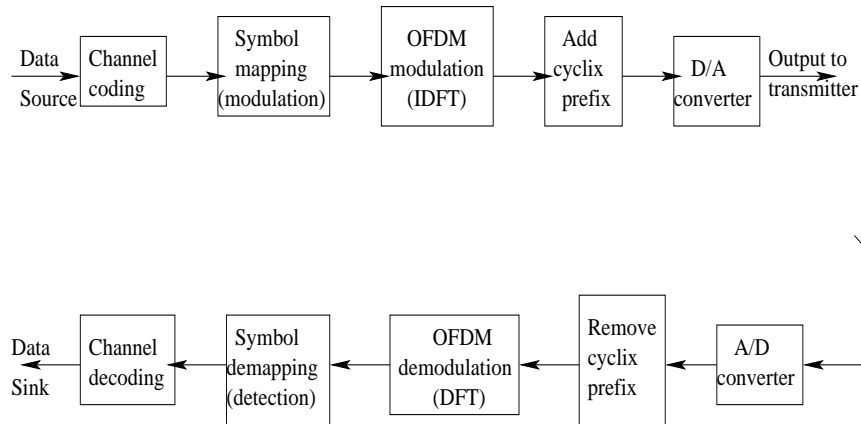


Figure 5.3: Block diagram of an OFDM system.

Its transfer function (Fourier transform) is

$$H(\omega) = \exp\left(-\frac{\alpha^2 \omega^2}{2}\right), \quad (5.8)$$

where $\alpha = \sqrt{2 \ln 2} T_b / 2\pi f_n$, T_b is the pulse coded modulation (PCM) bit time, and f_n is the channel bandwidth normalized to the PCM data rate.

OFDM modulation [71] is considered. For completeness, a brief introduction on OFDM is presented below. A basic block diagram of an OFDM system is shown in Figure 5.3. The information sequence is subdivided into K groups (subchannels). For coded OFDM systems, information bits are first passed through a channel encoder for each subchannel. The OFDM modulator consists of K independent subchannels. K signal points, X_k , are chosen from certain constellations according to any phase shift keying (PSK) or quadrature amplitude modulation (QAM) signaling set (symbol mapping). BPSK modulation is tested in our experiments. Those symbols can be viewed as values of the discrete Fourier transform

(DFT) of an OFDM signal $x(t)$. A cyclic prefix is appended to avoid intersymbol interference (ISI). The signal is transmitted through some channel after D/A conversion. The received signal may be expressed as $r(t) = x(t) \star h(t) + n(t)$, where \star represents convolution, and $n(t)$ is additive noise corrupting the signal. At the receiver end, inverse procedures are taken to recover the original information symbols, X_k . For a coded system, the output bits are decoded by a channel decoder.

In our experiments, we employ the GI-POF channel. The non-flat shape of the transfer function of the POF channel causes the subchannels to have different channel gains. In particular, the signal $x(t)$ is convolved with the impulse response of the channel (see Equation 5.7). Equivalently, the subchannel symbols X_k are multiplied by the transfer function of the channel (see Equation 5.8). We assume the additive noise in the system is AWGN, and SNR is denoted by γ_0 . From Equation 5.8, the gain ρ_k of each subchannel can be estimated. More importantly, the equivalent SNR of each subchannel can be estimated as $\gamma_k = \rho_k^2 \gamma_0$. The same image sources are used as in the previous example. In the experiments, the OFDM system has 128 subchannels. A channel SNR of $\gamma_0 = 3\text{dB}$ and $f_n = 1.3$ are selected. The bit rate of the system is 2.5 Gb/s, and a total of 256 packets are transmitted. The results are listed in Table 5.2. Similar conclusions can be drawn to those of the previous example. The total gain from multiplexing and UEP is about 2.3 dB.

Table 5.2: Average MSE and PSNR for 1.0 bpp/source and GI-POF channel.

	Scheme I		Scheme II		Scheme III		Scheme IV	
	MSE	PSNR	MSE	PSNR	MSE	PSNR	MSE	PSNR
Lena	41.16	31.99	55.07	30.72	12.04	37.33	12.79	37.06
Goldhill	67.81	29.82	89.83	28.60	27.15	33.79	31.73	33.12
Whitehouse	158.07	26.14	208.69	24.94	193.74	25.26	232.37	24.47
Lighthouse	172.15	25.77	205.70	25.00	419.93	21.90	477.37	21.34
Average	109.80	27.72	139.82	26.68	163.21	26.00	188.56	25.38

5.4 Conclusion

In this chapter, a multi-channel rate allocation algorithm is proposed. A fast algorithm assigns unequal error protection to each segment of a source to maximize the expected value of the correctly received source length. Differences between subchannels are utilized. The algorithm can be applied to many systems that suffer from nonideal channels. Applications to JPEG2000 transmission shows the superiority of the proposed algorithm by exploiting the differences between subchannels instead of using the average subchannel SNR. Significant UEP gains are achieved over EEP schemes. The rate allocation algorithm is independent of the transmitted source, but additional performance gains can be obtained by multiplexing multiple sources prior to rate allocation.

5.5 Proofs

5.5.1 Proof of Lemma 1

$$\begin{aligned}
& E_N[(r_{k_1}, s_{k_1}), \dots, (r_{k_N}, s_{k_N})] \\
&= l(r_{k_1}) \sum_{i=1}^N P_i(R) + \sum_{i=2}^N P_i(R) \sum_{j=2}^i l(r_{k_j}) \\
&= l(r_{k_1})(1 - p(r_{k_1}, s_{k_1})) + \sum_{i=2}^N P_i(R) \sum_{j=2}^i l(r_{k_j}) \\
&= l(r_{k_1})(1 - p(r_{k_1}, s_{k_1})) + (1 - p(r_{k_1}, s_{k_1})) \\
&\quad \times E_{N-1}[(r_{k_2}, s_{k_2}), \dots, (r_{k_N}, s_{k_N})].
\end{aligned}$$

5.5.2 Proof of Lemma 2

$$\begin{aligned}
& E_N[(r_1^*, s_1^*), (r_2^*, s_2^*), \dots, (r_N^*, s_N^*)] \\
&\geq (1 - p(r_{k_1}, s_{k_1})) [l(r_{k_1}) + E_{N-1}[(r_2^*, s_2^*), \dots, (r_N^*, s_N^*)]] \\
&\geq (1 - p(r_{k_1}, s_{k_1})) [l(r_{k_1}) + E_{N-1}[(r_{k_2}, s_{k_2}), \dots, (r_{k_N}, s_{k_N})]] \\
&= E_N[(r_{k_1}, s_{k_1}), (r_{k_2}, s_{k_2}), \dots, (r_{k_N}, s_{k_N})].
\end{aligned}$$

5.5.3 Proof of Corollary 1

$$\begin{aligned}
& E_N[(r_1^*, s_1^*), (r_2^*, s_2^*), \dots, (r_N^*, s_N^*)] \\
&\geq E_N[(r_1^*, s_1^*), (r_{k_2}, s_{k_2}), \dots, (r_{k_N}, s_{k_N})],
\end{aligned}$$

which implies

$$\begin{aligned}
& E_1[(r_1^*, s_1^*)] + (1 - p(r_1^*, s_1^*))E_{N-1}[(r_2^*, s_2^*), \dots, (r_N^*, s_N^*)] \\
& \geq E_1[(r_1^*, s_1^*)] + (1 - p(r_1^*, s_1^*)) \\
& \quad \times E_{N-1}[(r_{k_2}, s_{k_2}), \dots, (r_{k_N}, s_{k_N})].
\end{aligned}$$

This in turn yields

$$\begin{aligned}
& E_{N-1}[(r_2^*, s_2^*), \dots, (r_N^*, s_N^*)] \\
& \geq E_{N-1}[(r_{k_2}, s_{k_2}), \dots, (r_{k_N}, s_{k_N})].
\end{aligned} \tag{5.9}$$

Similarly, we can get

$$\begin{aligned}
& E_{N-i}[(r_{i+1}^*, s_{i+1}^*), \dots, (r_N^*, s_N^*)] \\
& \geq E_{N-i}[(r_{k_{i+1}}, s_{k_{i+1}}), \dots, (r_{k_N}, s_{k_N})].
\end{aligned} \tag{5.10}$$

5.5.4 Proof of Corollary 2

From *Corollary 1*, we have

$$E_2[(r_{N-1}^*, s_{N-1}^*), (r_N^*, s_N^*)] \geq E_2[(r_{N-1}^*, s_N^*), (r_N^*, s_N^*)]. \tag{5.11}$$

By Equation 5.6, we obtain

$$\begin{aligned}
& E_1[(r_{N-1}^*, s_{N-1}^*)] + (1 - p(r_{N-1}^*, s_{N-1}^*))E_1[(r_N^*, s_N^*)] \\
& \geq E_1[(r_{N-1}^*, s_N^*)] + (1 - p(r_{N-1}^*, s_N^*))E_1[(r_N^*, s_N^*)],
\end{aligned}$$

which indicates that

$$\begin{aligned}
& E_1[(r_{N-1}^*, s_N^*)] - E_1[(r_{N-1}^*, s_{N-1}^*)] \\
& \leq (p(r_{N-1}^*, s_N^*) - p(r_{N-1}^*, s_{N-1}^*))E_1[(r_N^*, s_N^*)].
\end{aligned} \tag{5.12}$$

If $s_N^* > s_{N-1}^*$, then Equation 5.12 has $RHS < 0$ and $LHS > 0$, which is a contradiction. Thus, we must have $s_{N-1}^* \geq s_N^*$. Similarly,

$$\begin{aligned} & E_3[(r_{N-2}^*, s_{N-2}^*), (r_{N-1}^*, s_{N-1}^*), (r_N^*, s_N^*)] \\ & \geq E_3[(r_{N-2}^*, s_{N-1}^*), (r_{N-1}^*, s_{N-1}^*), (r_N^*, s_N^*)], \end{aligned}$$

which yields

$$\begin{aligned} & E_1[(r_{N-2}^*, s_{N-2}^*)] + (1 - p(r_{N-2}^*, s_{N-2}^*)) \\ & \times E_2[(r_{N-1}^*, s_{N-1}^*), (r_N^*, s_N^*)] \geq E_1[(r_{N-2}^*, s_{N-1}^*)] \\ & + (1 - p(r_{N-2}^*, s_{N-1}^*))E_2[(r_{N-1}^*, s_{N-1}^*), (r_N^*, s_N^*)]. \end{aligned}$$

This leads to

$$\begin{aligned} & E_1[(r_{N-2}^*, s_{N-1}^*)] - E_1[(r_{N-2}^*, s_{N-2}^*)] \\ & \leq (p(r_{N-2}^*, s_{N-1}^*) - p(r_{N-2}^*, s_{N-2}^*)) \tag{5.13} \\ & \times E_2[(r_{N-1}^*, s_{N-1}^*), (r_N^*, s_N^*)]. \end{aligned}$$

By the same argument as above, we obtain $s_{N-2}^* \geq s_{N-1}^*$. The same method can be used to show that $s_{i-1}^* \geq s_i^*$, for $2 \leq i \leq N$.

REFERENCES

- [1] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, vol. 27, pp. 379–423, July 1948.
- [2] J. M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients,” *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3445–3462, Dec. 1993.
- [3] A. Said and W. A. Pearlman, “A new, fast, and efficient image codec based on set partitioning in hierarchical trees,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, June 1996.
- [4] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image compression fundamentals, standards and practice*. Boston: Kluwer Academic Publishers, 2002.
- [5] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error-correcting coding and decoding: Turbo codes,” in *Proceedings, IEEE International Conference on Communications*, vol. 2, Geneva, Switzerland, May 1993, pp. 1064–1070.
- [6] D. J. C. MacKay and R. Neal, “Good codes based on very sparse matrices,” in *Proceedings, 5th IMA Conference Cryptography and Coding*, Berlin, Germany, 1995, Springer Lecture Notes in Computer Science.
- [7] R. G. Gallager, “Low-density parity-check codes,” *IRE Transactions on Information Theory*, vol. IT-8, pp. 21–28, Jan. 1962.
- [8] ———, *Information Theory and Reliable Communication*. Wiley, New York, 1968.
- [9] D. J. C. MacKay, “Good error-correcting codes based on very sparse matrices,” *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
- [10] N. Alon and M. Luby, “A linear time erasure-resilient code with nearly optimal

- recovery,” *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1732–1736, Nov. 1996.
- [11] R. M. Tanner, “A recursive approach to low-complexity codes,” *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, Sep. 1981.
- [12] I. Kozintsev, J. Chou, and K. Ramchandran, “Image transmission using arithmetic coding based continuous error detection,” in *Proceedings, Data Compression Conference*, Snowbird, Utah, 1998, pp. 339–348.
- [13] B. Steingrimsson, J. Moon, and T. Oenning, “Signal space detection for DVD optical recording,” *IEEE Transactions on Magnetics*, vol. 37, no. 2, pp. 670–675, Mar. 2001.
- [14] Z. Peng, Y. Huang, and D. J. Costello, “Turbo codes for image transmission — a joint channel and source decoding approach,” *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 868–879, Jun. 2000.
- [15] T. Guionnet and C. Guillemot, “Soft decoding and synchronization of arithmetic codes: Application to image transmission over noisy channels,” *IEEE Transactions on Image Processing*, vol. 12, no. 12, pp. 1599–1609, Dec. 2003.
- [16] B. Banister, B. Belzer, and T. Fischer, “Robust image transmission using JPEG2000 and Turbo-codes,” *IEEE Signal Processing Letters*, vol. 9, no. 4, pp. 117–119, Apr. 2002.
- [17] C. Lan, T. Chu, K. R. Narayanan, and Z. Xiong, “Scalable image and video transmission using irregular repeat-accumulate codes with fast algorithm for optimal unequal error protection,” *IEEE Transactions on Communication*, vol. 52, no. 7, pp. 1092–1101, Jul. 2004.
- [18] X. Pan, A. Cuhadar, and A. H. Banihashemi, “Combined source and channel coding with JPEG2000 and rate-compatible low-density parity-check codes,” *IEEE Transactions on signal processing*, vol. 54, no. 3, pp. 1160–1164, Mar. 2006.
- [19] C. Zhong and J. P. Havlicek, “LDPC codes for robust transmission of images over wireless channels,” in *Conference Record of the 35th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, California, Nov. 2001, pp. 797–800.

- [20] *www.kakadusoftware.com*.
- [21] I. B. Djordjevic, S. Sundararajan, and B. Vasic, "Projective-plane iteratively decodable block codes for WDM high-speed long-haul transmission systems," *Journal of Lightwave technology*, vol. 22, no. 3, pp. 695–702, Mar. 2004.
- [22] B. Vasic and O. Milenkovic, "Combinatorial constructions of low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 50, no. 6, pp. 1156–1176, June 2004.
- [23] P. G. Sherwood and K. Zeger, "Progressive image coding for noisy channels," *IEEE Signal Processing Letters*, vol. 4, no. 7, pp. 189–191, Jul. 1997.
- [24] A. Mohr, E. Riskin, and R. Ladner, "Graceful degradation over packet erasure channels through forward error correction," in *Proceedings, Data Compression Conference*, Snowbird, UT, Mar. 1999, pp. 92–101.
- [25] Z. Wu, A. Bilgin, and M. W. Marcellin, "Joint source/channel coding for image transmission with JPEG2000 over memoryless channels," *IEEE Transactions on Image Processing*, vol. 14, no. 8, pp. 1020–1032, 2005.
- [26] V. Chande and N. Farvardin, "Joint source-channel coding for progressive transmission of embedded source coders," in *Proceedings, Data Compression Conference*, Snowbird, UT, Mar. 1999, pp. 52–61.
- [27] V. Stankovic, R. Hamzaoui, and D. Saupe, "Fast algorithm for rate-based optimal error protection of embedded codes," *IEEE Transactions on Communications*, vol. 51, no. 11, pp. 1788–1795, Nov. 2003.
- [28] R. Hamzaoui, V. Stankovic, and Z. Xiong, "Rate-based versus distortion-based optimal joint source-channel coding," in *Proceedings, Data Compression Conference*, Snowbird, UT, Apr. 2002, pp. 63–72.
- [29] C. Lan, K. R. Narayanan, and Z. Xiong, "Source-optimized irregular repeat accumulate codes with inherent unequal error protection capabilities and their application to scalable image transmission," *IEEE Transactions on Image Processing*, vol. 15, no. 7, pp. 1740–1750, July 2006.
- [30] R. Hamzaoui, V. Stankovic, and Z. Xiong, "Optimized error protection of scalable image bit streams [advances in joint source-channel coding for images],"

IEEE Signal Processing Magazine, vol. 22, no. 6, pp. 91–107, 2005.

- [31] F. MacWilliams and N. Sloane, *The theory of error-correcting codes*. North Holland Publishing Company, 1977.
- [32] V. Kumar and O. Milenkovic, “Unequal error protection LDPC codes based on Plotkin-type constructions,” in *Proceedings, Globecom 2004*, Dallas, Dec. 2004.
- [33] E. Eleftheriou and S. Ölçer, “Rate-compatible low-density parity-check codes for digital subscriber lines,” in *Proceedings, IEEE International Conference on Communications*, vol. 1, Jun. 2004, pp. 415–419.
- [34] *DCI Digital cinema system specification V1.0*, <http://www.dcmovies.com>.
- [35] <http://www.wimedia.org>.
- [36] I. Dumer, “Recursive decoding and its performance for low-rate Reed-Muller codes,” *IEEE Transactions on Information theory*, vol. 50, no. 5, pp. 811–823, May 2004.
- [37] V. Kumar and O. Milenkovic, “On unequal error protection LDPC codes based on Plotkin-type constructions,” *IEEE Transactions on Communications*, vol. 54, no. 6, pp. 994–1005, Jun. 2006.
- [38] E. Eleftheriou and S. Ölçer, “Low-density parity-check codes for digital subscriber lines,” in *Proceedings, IEEE International Conference on Communications*, vol. 3, Apr. 2002, pp. 1752–1757.
- [39] M. Blaum, P. Farrell, and H. V. Tilborg, *Handbook of Coding Theory*. Elsevier, 1998, ch. Array codes.
- [40] J. A. C. Bingham, “Multicarrier modulation for data transmission: An idea whose time has come,” *IEEE Communications Magazine*, vol. 28, no. 5, pp. 5–14, May 1990.
- [41] I. Kalet, “The multitone channel,” *IEEE Transactions on Communications*, vol. 37, no. 2, pp. 119–124, Feb. 1989.
- [42] D. Hughes-Hartogs, “Ensemble modem structure for imperfect transmission media,” u.S. Patents Nos. 4,679,227 (July 1987), 4,731,816 (Mar. 1988), and

4,833,706 (May 1989).

- [43] P. S. Chow, J. M. Cioffi, and J. A. C. Bingham, "A practical discrete multi-tone transceiver loading algorithm for data transmission over spectrally shaped channels," *IEEE Transactions on Communications*, vol. 43, no. 234, pp. 773–775, Feb. 1995.
- [44] R. F. H. Fischer and J. B. Huber, "A new loading algorithm for discrete multitone transmission," in *Proceedings, Global Telecommunications Conference*, Nov. 1996, pp. 724–728.
- [45] B. S. Krongold, K. Ramchandran, and D. L. Jones, "Computationally efficient optimal power allocation algorithms for multicarrier communication systems," *IEEE Transactions on Communications*, vol. 48, no. 1, pp. 23–27, Jan. 2000.
- [46] S. S. Pradhan and K. Ramchandran, "Optimized embedded multicarrier modulation for efficient delivery of layered video data," in *Proceedings, IEEE International Conference on Communications*, vol. 2, June 1998, pp. 1009–1012.
- [47] P. J. Cherriman, T. Keller, and L. Hanzo, "Subband-adaptive turbo-coded OFDM-based interactive video telephony," *IEEE Transactions on Circuits, Systems, Video Technology*, vol. 12, no. 10, pp. 829–839, Oct. 2002.
- [48] J. Song and K. J. R. Liu, "Robust progressive image transmission over OFDM systems using space-time block code," *IEEE Transactions on Multimedia*, vol. 4, no. 3, pp. 394–406, Sep. 2002.
- [49] Y. Sun, Z. Xiong, and X. Wang, "Scalable image transmission over differentially space-time coded OFDM systems," in *Proceedings, Global Telecommunications Conference*, Nov. 2002, pp. 379–383.
- [50] Y. Sun and Z. Xiong, "Progressive image transmission over space-time coded OFDM-based MIMO systems with adaptive modulation," *IEEE Transactions on Mobile Computing*, vol. 5, no. 8, pp. 1016–1028, Aug. 2006.
- [51] C. Kuo, C. Kim, and C. J. Juo, "Robust video transmission over wideband wireless channel using space-time coded OFDM systems," in *IEEE Wireless Communications and Networking Conference*, March 2002, pp. 931–936.
- [52] Y. S. Chan, P. C. Cosman, and L. B. Milstein, "A cross-layer diversity technique

- for multicarrier OFDM multimedia networks,” *IEEE Transactions on Image Processing*, vol. 15, no. 4, pp. 833–847, Apr. 2006.
- [53] H. Zheng and K. J. R. Liu, “Robust image and video transmission over spectrally shaped channels using multicarrier modulation,” *IEEE Transactions on Multimedia*, vol. 1, no. 1, pp. 88–103, Mar. 1999.
- [54] S. Zhao, Z. Xiong, and X. Wang, “Optimal resource allocation for wireless video over CDMA networks,” *IEEE Transactions on Mobile Computing*, vol. 4, no. 1, pp. 56–67, Jan./Feb. 2006.
- [55] S. Zhao, Z. Xiong, X. Wang, and J. Hua, “Progressive video delivery over wide-band wireless channels using space-time differentially coded OFDM systems,” *IEEE Transactions on Mobile Computing*, vol. 5, no. 4, pp. 303–316, Apr. 2006.
- [56] V. Stankovic, R. Hamzaoui, and D. Saupe, “Fast algorithm for rate-based optimal error protection of embedded codes,” *IEEE Transactions on Communications*, vol. 51, no. 11, pp. 1788–1795, Nov. 2003.
- [57] “Radio broadcast systems, Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers,” eTS 300 401 (1994).
- [58] “Digital broadcasting systems for television, sound and data services, framing structure, channel coding and modulation for digital terrestrial television,” eTS 300 744 (1997).
- [59] F. Mederer, R. Jäger, P. Schnitzer, H. Unold, M. Kicherer, K. J. Ebeling, M. Naritomi, and R. Yoshida, “Multi-Gb/s graded-index POF data link with butt-coupled single-mode InGaAs VCSEL,” *IEEE Photonics Technology Letters*, vol. 12, no. 2, pp. 199–201, Feb. 2000.
- [60] T. Matsuoka, T. Ito, and T. Kaino, “First plastic optical fibre transmission experiment using 520nm LEDs with intensity modulation/direct detection,” *IEE Electronics Letters*, vol. 36, no. 22, pp. 1836–1837, Oct. 2000.
- [61] <http://www.technologynewsdaily.com/node/2927>.
- [62] S. Randel, S. C. J. Lee, B. Spinnler, F. Breyer, H. Rohde, J. Walewski, A. M. J. Koonen, and A. Kirstädter, “1 Gbit/s transmission with 6.3 bit/s/hz spectral efficiency in a 100 m standard 1 mm step-index plastic optical fibre link using

- adaptive multiple sub-carrier modulation,” in *32nd European Conference on Optical Communication*, Sep. 2006, pp. 41–42.
- [63] R. Hamzaoui, V. Stankovic, and Z. Xiong, “Rate-based versus distortion-based optimal joint source-channel coding,” in *Proceedings, Data Compression Conference*, Snowbird, Utah, April 2002, pp. 63–72.
- [64] V. Chande and N. Farvardin, “Progressive transmission of images over memoryless noisy channels,” *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 6, pp. 850–860, Jun. 2000.
- [65] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, “A soft-input soft-output maximum a posteriori (MAP) module to decode parallel and serial concatenated codes,” *TDA Progress Report*, pp. 42–127, Nov. 1996.
- [66] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.
- [67] D. N. Bowitch and L. B. Milstein, “On the performance of hybrid FEC/ARQ systems using rate compatible punctured Turbo (RCPT) codes,” *IEEE Transactions on Communications*, vol. 48, no. 6, pp. 948–959, Jun. 2000.
- [68] T. Chuu, Z. Liu, Z. Xiong, and X. Wu, “Joint UEP and layered source coding with application to transmission of JPEG2000 coded images,” in *Global Comm. Conf.*, vol. 3, 2001, pp. 2036–2039.
- [69] M. Kalman and B. Girod, “Optimal channel-time allocation for the transmission of multiple video streams over a shared channel,” in *IEEE 7th workshop on multimedia signal processing*, Oct. 2005, pp. 1–4.
- [70] B.-G. Shin, J.-H. Park, and J.-J. Kim, “Low-loss, high-bandwidth graded-index plastic optical fiber fabricated by the centrifugal deposition method,” *Applied Physics Letters*, vol. 82 (26), pp. 4645–4647, Jun. 2003.
- [71] R. Prasad, *OFDM for wireless communications systems*. Artech House, Inc., 2004.