TECHNIQUES FOR UNEQUAL ERROR PROTECTION

HO MAN-SHING

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF PHILOSOPHY

IN

INFORMATION ENGINEERING

JULY 2001

摘 要

本論文提出了一種低複雜度的通道編碼分配演算法，在高噪通道中傳送 SPIHT（Set Partitioning In Hierarchical Trees）編碼格式的圖像時提供不均勻差錯保護(Unequal Error Protection) ，用不同截斷速率的 RCPC（Rate-compatible Punctured Convolutional）編碼爲每個 SPIHT 編碼格式的資料包提供不均勻差錯保護，同時在解碼器中用列舉式維特比演算法（List Viterbi Algorithm）來減少端到端的圖像傳送失真。這種新的編碼分配演算法能在很大的傳送速率範圍內提供很好的峰值信噪比 PSNR（Peak Signal to Noise Ratio）。

此外，基於格子編碼調製 TCM（Trellis-coded Modulation），本文另提出一種不均勻差錯保護編碼，用精細（Fine）和粗糙（Coarse）星座圖（Constellation）爲資料中具有兩種不同重要性的資訊類別提供不均勻的差錯保護。利用這種編碼技術，在編碼器端交替地使用格子碼的精細和粗糙星座圖將資料進行編碼；在解碼器端，用維特比演算法來尋找最接近的碼序列作爲解碼輸出。在相同編碼速率下，不均勻差錯保護編碼技術能爲重要的資料提供更好的保護，及達致更低的錯誤概率。

Acknowledgement

First of all, I would like to express my sincere gratitude to my supervisor, Prof. Keang-Po Ho, for his guidance. In the period of my study, He gave me help, support, encouragement and guided me to the right research direction. I am inspired by his profound thinking, depth of knowledge and nice personality. I have learnt a lot from him on doing research. I would also like to thank my co-supervisor, Prof. Wai-Ho Yeung for his help and support.

I am grateful to my friends Kwong Cheuk Fai, Chen Chung Shue and Tse Yan Tung, who had a lot of stimulating discussions with me on various topics in unequal error protection, combined source-channel coding and communication theory.

Finally, I would like to thank my family for their love and endless support throughout all these years.

Abstract

A simple and low complexity algorithm is derived for channel code allocation to provide unequal error protection for set-partitioning-in-hierarchical trees (SPIHT) coded image transmitted over noisy channels. Rate–compatible punctured convolutional (RCPC) codes with different punctured rate are used to provide unequal error protection for each SPIHT-coded data packet and list Viterbi algorithm (LVA) is used in the decoder to minimize the end-to-end image distortion. The new code allocation algorithm can achieve better peak signal-to-noise ratio (PSNR) at a wide range of transmission rate.

A new unequal error protection code based on trellis-coded modulation (TCM) is invented to provide unequally error-protected information with two levels of importance. Unequal error protection is provided by using fine and coarse constellations. The fine and coarse constellations are used alternatively to transmit the data and Viterbi algorithm is used in the decoder to find the best sequence as decoded output. The unequal error protection code can provide more protection for the important data. Probability of error for the important data is lower than equal protection scheme for the same code rate.

List of Abbreviation

AM	Amplitude Modulation
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BSC	Binary Symmetric Channel
CRC	Cyclic Redundancy Code
EZW	Embedded Zero Wavelet
LSB	Least Significant Bit
LVA	List Viterbi Algorithm
MSB	Most Significant Bit
PLVA	Parallel List Viterbi Algorithm
PSNR	Peak Signal to Noise Ratio
RCPC	Rate Compatible Punctured Convolutional
SPIHT	Set Partitioning In Hierarchical Trees
TCM	Trellis Coded Modulation
UEP	Unequal Error Protection
VA	Viterbi Algorithm

Contents

Acknowledgement .... **i**

Abstract .... **ii**

List of Abbreviation .... **iii**

1. Introduction **1**

1.1 Digital Communication System 3

1.2 Thesis Organization 4

2. Error-Correcting Codes **6**

2.1 Convolutional Codes 7

2.1.1 Generator Polynomials 8

2.1.2 Generator Matrix 9

2.1.3 Circuit Diagram 10

2.1.4 State-transition Diagram 11

2.1.4 Trellis Diagram 12

2.1.5 Distance property 13

2.2 Rate-Compatible Punctured Convolutional Codes 14

2.3 Trellis-Coded Modulation 17

2.3.1 General Model of TCM 18

2.3.2 Trellis Representation 20

2.3.3 Set Partitioning 21

2.3.4 Code Modulation 23
2.4 Decoding Algorithm 25
2.4.1 Viterbi Algorithm 27
2.4.2 List Viterbi Algorithm 30

3. Unequal-Error-Protection for Embedded Image Coder 33
3.1 SPIHT Coder 35
3.1.1 Progressive Image Transmission 36
3.1.2 Set Partitioning Sorting Algorithm 37
3.1.3 Spatial Orientation Trees 38
3.2 System Description 39
3.3 Code Allocation 41
3.4 System Complexity 42
3.5 Simulation Result 43

4. Unequal-Error-Protection Provided by Trellis-Coded Modulation 51
4.1 System Description 52
4.2 Unequal Constellation 53
4.3 Free Distance 55
4.4 Simulation Results 59

5. Conclusion 63

Bibliography 65

Chapter 1

Introduction

In a communication system, the transmitted data passing through the channel can be corrupted by the unavoidable noise and other imperfections. With the noisy signal, the receiver may decode the original transmitted data with certain error probability. Error correction codes can provide a certain level of error protection to these corrupted data. In error correction codes, some redundant symbols are added "intelligently" to the data by the error-control encoder of the system. These redundancy symbols are used to correct the erroneous data at the receiver.

The transmitted data is often assumed to be equally significant. However, the significance of data may be taken into account to achieve improvement in the overall performance. With a properly designed system, an error correcting code can provide more protection for selected data and less protection for others. Such codes are called *unequal error protection codes* (UEP codes). UEP codes are families of error-control codes that can change their coding strengths flexibly and adaptively. UEP codes are useful for protecting data sources such as audio, image, and video that are typically able to be compressed with unequal importance of bits. As an example, the data bits can be divided into two importance classes: more significant bits (MSB) and less significant bits (LSB). Without UEP, the two classes achieve the same raw bit error probability. The purpose of unequal error protection is to protect the most important bits with a lower bit error probability but the less important bits with higher error probability.

The emphasis of this thesis is methods to provide unequal error protection. In the first part of the thesis, we design an algorithm to provide unequal error protection for an image coder. A simple and low complexity algorithm is derived for channel codes allocation to provide unequal error protection for set-partitioning-in-hierarchical trees (SPIHT) coded image transmitted over noisy channels. Rate–compatible punctured convolutional (RCPC) codes with different punctured rate are used to provide unequal error protection for each SPIHT coded data packet and list Viterbi algorithm (LVA) is used in the decoder to minimize the end-to-end image distortion.

A new unequal error protection code based on trellis-coded modulation (TCM) is also studied. The new TCM scheme can unequally protect information with two levels of importance. Fine and coarse constellations are adopted to achieve unequal error protection. The fine and coarse constellations are used alternatively to transmit the source data. Viterbi algorithm is used as decoder to find the best sequence as decoded output. Overall performance of the UEP code in term of probability of error is only slightly worse than the performance of equal protection scheme for the same code rate but can provide more protection for the important data. The probability of error for the important data is lower than equal protection scheme. Detailed descriptions of the technical issues will be carefully examined in the latter chapters.

1.1 Digital Communication System

Figure 1.1 shows a block diagram of a point-to-point digital communication system. The data to be transmitted through this system can come from some analog source, in which case it must first be converted into digital format (digitized), or a digital information source. Ideally, the digital data is processed by source encoder to remove unnecessary redundancy from the source data, i.e., the source data is

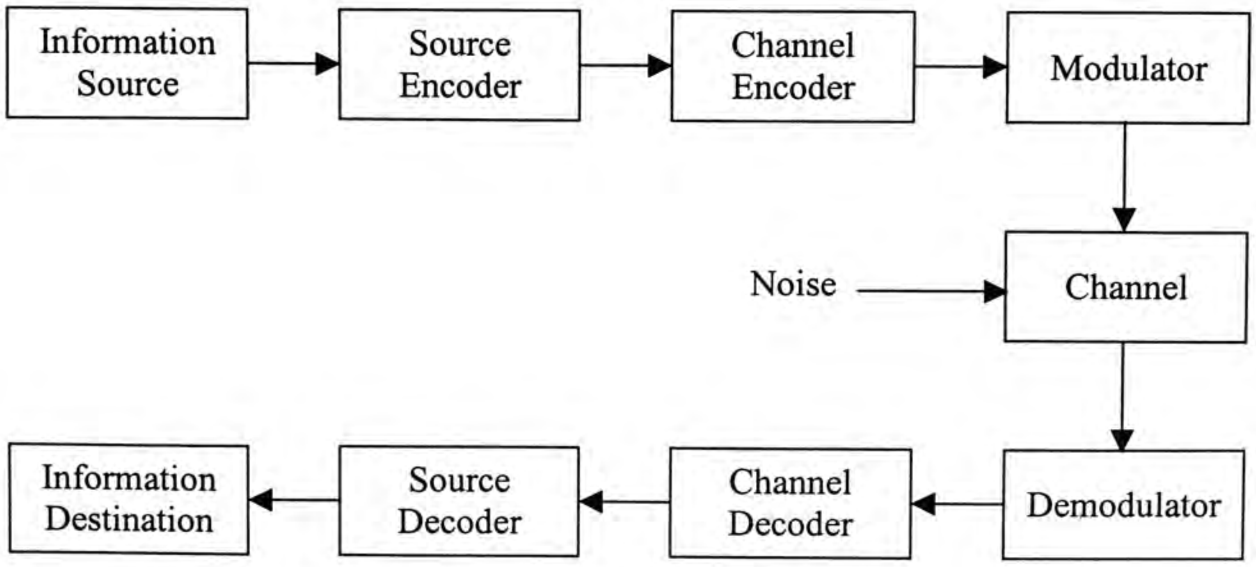


Figure 1.1: Block diagram of digital communication system

compressed. The channel encoder operates on this compressed data and introduces controlled redundancy for transmission. The modulator converts the discrete symbols into waveform, which are transmitted through channel. The demodulator reconverts the waveforms into a discrete sequence of received symbols and the channel decoder reproduces an estimate of the compressed input data sequence, which is subsequently reconverted into the original signal or data sequence by source decoder.

To provide unequal error protection for information source in the above system, the most obvious way is to group them with respect to their significance inside the source encoder. According to their significant levels, we can code each level separately inside the channel encoder by using corresponding error controlling codes.

1.2 Thesis Organization

Chapter 2 gives a brief summary and background information about convolutional codes, rate-compatible punctured convolutional codes (RCPC codes) and trellis codes. Besides, we also discuss their decoding algorithm, Viterbi algorithm and list Viterbi algorithm.

A simple and low complexity algorithm for channel codes allocation is derived in chapter 3 to provide unequal error protection for SPIHT-coded image transmitted over noisy channels. Detailed channel coding allocation algorithm, system complexity and simulation result are presented and discussed.

In chapter 4, a new unequal error protection code for TCM is proposed to provide unequally protect information with two levels of importance. The free distances for this new code using amplitude modulation (AM) scheme is simulated. Furthermore, the code performance in term of probability of error over AWGN channel is also presented

Finally, a conclusion is provided in chapter 5.

Chapter 2

Error-Correcting Codes

Error-correcting codes are used to provide reliable transmission of digital information over a channel. The redundancy symbols, which are added to source data in the channel encoder, are used to correct the possible erroneous data at the received side. There are two basic types of error-correcting codes, block code and sliding-window codes.

In the block coding, the encoder accepts a k-bit message block and generated an n-bit *codeword*. The codewords are produced on a block-by-block basis. The encoding of each block is independent from the previous blocks. In other words, there exists no memory from one block to another block.

For the sliding window coding, the information bits are divided into blocks of length k, and each block is encoded into n-bit codeword. However, this n-bit codeword is not only determined by present block, but also by previous λ blocks. In other words, the encoding of each block of k information bits into a block of n coded bits is dependent on a sliding window of $k(\lambda+1)$ information bits.

This thesis focuses on two sliding window codes, convolutional codes and trellis codes to provide unequal error protection to unequally important information.

2.1 Convolutional Codes

Convolutional code is one kind of sliding window codes first introduced by Elias [1] in 1955 as an alternative to block codes. The advances in various decoding algorithms for convolutional codes like sequential decoding proposed by Wozencraft [2], maximum likelihood decoding scheme proposed by Viterbi [3], spawned many practical applications of convolutional codes to digital transmission over both wire and wireless channels.

Convolutional codes differ from block codes in which the encoder contains memory. A (n, k, λ) convolutional code will generate n encoded bits for each k information bits with λ stages of memory. Thus, there are $(n\text{-}k)$ redundant bits added to each k information bits. Typically, n and k should be small integers and $k < n$, but the memory size λ must be made large in order to achieve low error probabilities. For special case, when $k = 1$, the information sequence need not be divided into blocks and can be processed continuously. Figure 2.1 shows the block diagram of convolutional code. Because n output bits will be generated for each k input bits by convolutional encoder, the *code rate* (R) of convolutional code is simply $R = k / n$.

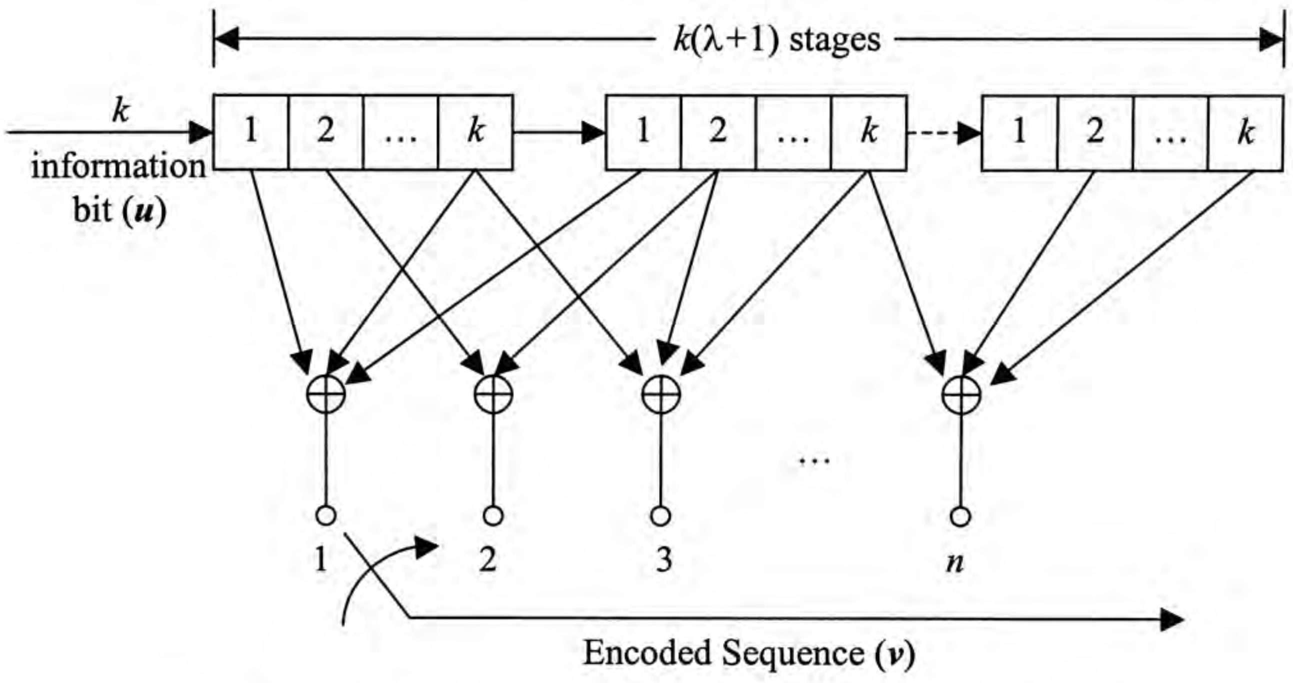

$k(\lambda+1)$ stages
k
information bit ($\boldsymbol{u}$)
1 2 ... k
1 2 ... k
1 2 ... k
...
1
2
3
n
Encoded Sequence ($\boldsymbol{v}$)

Figure 2.1: The block diagram of (n, k, λ) convolutional encoder

As discussed in next sections, there are several equivalent ways of specifying a convolutional code, including generator polynomials, generator matrix, circuit diagram, state-transition diagram, and trellis diagram.

2.1.1 Generator Polynomials

The encoder of a convolutional code can be described by relating the input and output sequences in a transform-domain over $GF(2)$ [4]. To accomplish this, let

$$u(D) = \sum_{i=0}^{\infty} u_i D^i \quad \text{or } \boldsymbol{u} = (u_0, u_1, u_2, \ldots) \tag{2.1}$$

be input sequence to the encoder and

$$v(D) = \sum_{i=0}^{\infty} v_i D^i \text{ or } \boldsymbol{v} = (v_0, v_1, v_2, \ldots), \tag{2.2}$$

be output sequence to the decoder, where the indeterminate D can be interpreted as the unit delay operator. The input-output relationships are expressed

$$v^{(i)}(D) = u(D) \times G_i(D), \tag{2.3}$$

where

$$G_i(D) = \sum_{j=0}^{\lambda} g_{i,j} D^j \tag{2.4}$$

are the *generator polynomials* of convolutional code. For a (n, k, λ) code, there are n generator polynomials. After a multiplexing of the output $v^{(i)}(D)$, the output codeword polynomial becomes

$$v(D) = \sum_{i=0}^{n-1} D^i v^{(i)}(D) \tag{2.5}$$

2.1.2 Generator Matrix

The convolutional code is linear such that the coding equation can be described by the matrix multiplication [4]. The generator polynomial can be organized into a semi-infinite matrix $\boldsymbol{G}$, which is called *generator matrix*. The input-output relationships are defined as

$$\boldsymbol{v} = \boldsymbol{uG} \tag{2.6}$$

where

$$G = \begin{bmatrix} G_0 & G_1 & G_2 & \cdots & G_\lambda & & & \\ & G_0 & G_1 & G_2 & \cdots & G_\lambda & & \\ & & G_0 & G_1 & G_2 & \cdots & G_\lambda & \\ & & & \ddots & & \cdots & & \ddots \end{bmatrix}, \tag{2.7}$$

and each G_l is a k by n matrix specified below:

$$G_l = \begin{bmatrix} g_{1,l}^{(1)} & g_{1,l}^{(2)} & \cdots & g_{1,l}^{(n)} \\ g_{2,l}^{(1)} & g_{2,l}^{(2)} & \cdots & g_{2,l}^{(n)} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,l}^{(1)} & g_{k,l}^{(2)} & \cdots & g_{k,l}^{(n)} \end{bmatrix}, \tag{2.8}$$

where $g_i^{(j)} = (g_{i,0}^{(j)}, g_{i,1}^{(j)}, g_{i,3}^{(j)}, \cdots g_{i,\lambda}^{(j)}), 1 \le i \le k, 1 \le j \le n$, are the *generator sequences*. The generator matrix is used to specify a convolutional code in chapter 3 later.

Convolutional codes also have parity check matrices, denoted ***H***, in which for any encoded sequence ***v***,

$$\boldsymbol{v}\boldsymbol{H}^T = \boldsymbol{0}, \tag{2.9}$$

where **0** denotes a sequence of zeros of arbitrary length.

2.1.3 Circuit Diagram

The convolutional encoder can be also implemented by linear sequential circuit. The circuits consist of a shift register with $k(\lambda+1)$ stages. At each instant of time, k information bits (***u***) enter the shift register and the contents of the last k states of the shift register dropped. After the k bits have entered the shift register, n linear combinations of the contents of the shift register are computed and used to generate the n-bits output codeword (***v***). From the above coding procedure, it is obvious that the n encoder outputs not only depend on the most recent k bits that have entered the encoder, but also on the $k\lambda$ contents of the first $k\lambda$ stages of the shift register before the k bits arrived. An example of circuit diagram of convolutional code is shown in the Figure 2.2. This example will be used in later sections of this chapter.

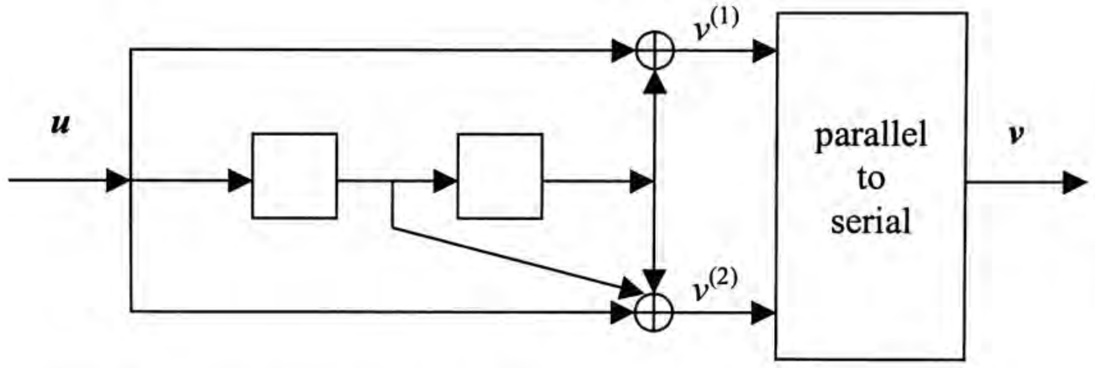


Figure 2.2: The circuit diagram of (2, 1, 2) convolutional encoder with generator polynomials of $g_1 = 1 + D^2$ and $g_2 = 1 + D + D^2$

2.1.4 State-transition Diagram

Because a convolutional encoder has finite memory, it can be specified completely by the *state-transition diagram* [4]. Figure 2.3 shows the state transition diagram of (2, 1, 2) convolutional encoder, which is shown in the Figure 2.2.
Figure 2.3 has $2^{k\lambda}$ states, corresponding to all possible contents of the $k\lambda$ bit storages. The state transition edges are labeled by $\boldsymbol{a}/\boldsymbol{b}$ where $\boldsymbol{a}$ corresponds to the k-bit input

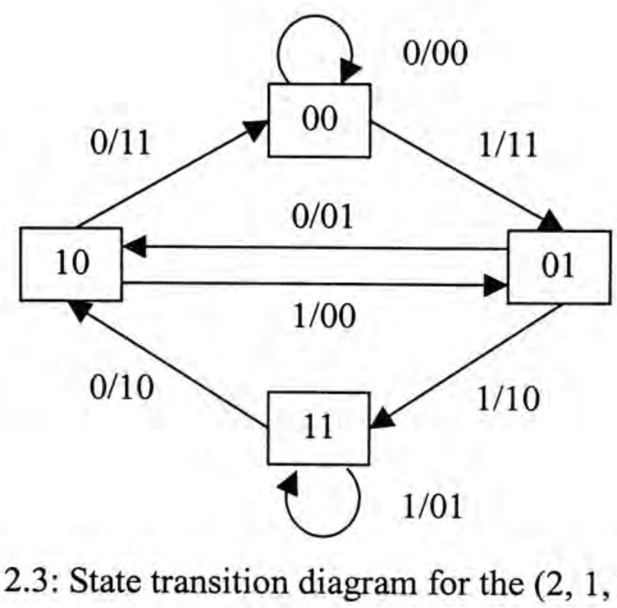


Figure 2.3: State transition diagram for the (2, 1, 2) encoder

and ***b*** corresponds to the n-bit output from the convolutional encoder. The number lines emerging from each state is equal to the number of possible input to the encoder at that state, which is equal to 2^k. The number of line merging at each state is equal to the number of states from which a transition is possible to that state, which is equal to the number of possible combinations of bits that leave the encoder as the k bits enter the encoder, i.e., 2^k.

2.1.4 Trellis Diagram

To show the various states and coded sequence as time evolves, it is traditional and instructive to describe all possible transitions of a convolutional code by *trellis diagram*. The trellis diagram can be obtained by specifying all $2^{k\lambda}$ states on a vertical axis and repeating this vertical axis along the time axis. Each transition from a state to another state is denoted by a line connecting the two states on two adjacent stages corresponding to two time instances. The trellis diagram has 2^k branches of the trellis leaving each state and 2^k branches merging at each. There is an edge from state S on stage t to state S' on stage t+1 if and only if there is an edge from state is labeled with encoder output corresponding to transition in state-transition diagram. Each path in the trellis diagram corresponds to a code sequence. Figure 2.4 shows the trellis diagram for the code described by the encoder of Figure 2.2. In the later chapter, trellis diagram is used to specify the encoder and decoder for detailed analysis.

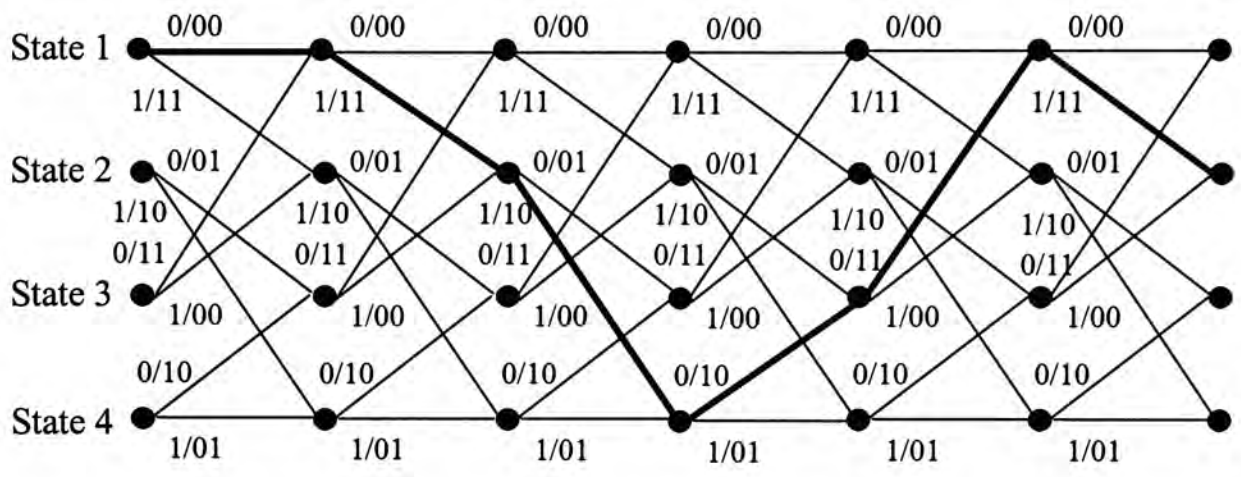


Figure 2.4: Trellis diagram for the encoder in Figure 2.2 with input sequence $\boldsymbol{u} = [0,1,1,0,0,1, \ldots]$ and encoded sequence $\boldsymbol{v} = [00, 11, 10, 10, 11, 11, \ldots]$

2.1.5 Distance property

The performance of a convolutional code depends on the decoding algorithm employed and the distance measures of the code [4]-[5]. Since the convolutional codes are a subclass of all linear codes, the set of Hamming distance between code sequences equals the set of distances of the coded sequences from the all-zero sequence. The minimum Hamming distance of convolutional codes sometimes is called the free Hamming distance and denoted by d_H, i.e.

$$\begin{aligned} d_H &= \min\{d(\vec{v}', \vec{v}''): \vec{v}' \neq \vec{v}''\} \\ &= \min\{d(\vec{v}, \vec{0}): \vec{v} \neq \vec{0}\} \\ &= \min\{w(\vec{v}): \vec{v} \neq 0\} \end{aligned} \tag{2.10}$$

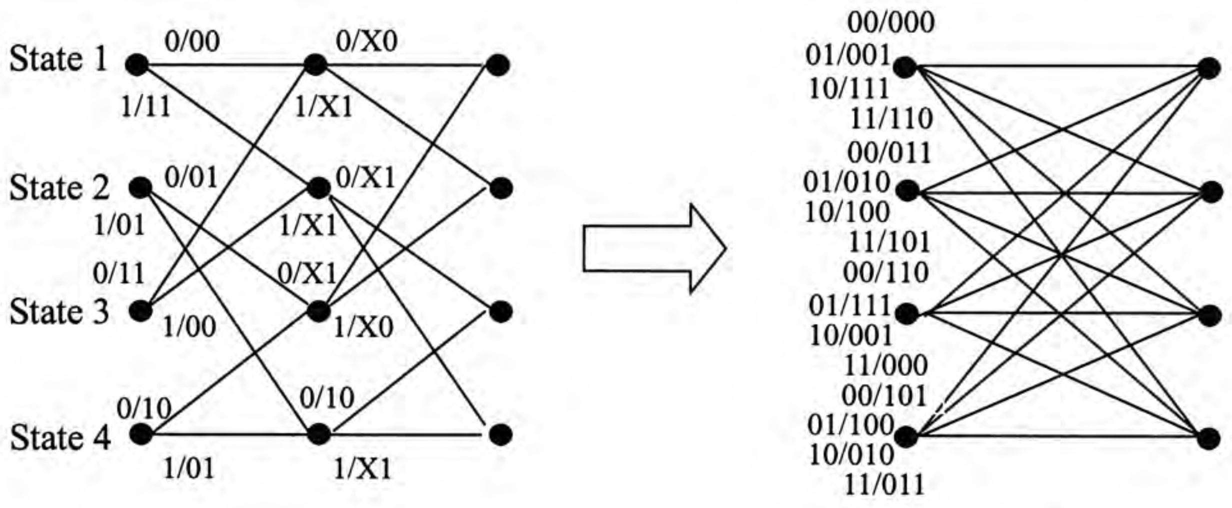
where $\vec{v}'$ and $\vec{v}''$ are the codeword corresponding to the information sequence $\vec{u}'$ and $\vec{u}''$ respectively. The free Hamming distance is minimum Hamming weight of codewords and can be computed by aid of trellis diagram. d_H is equal to minimum

Hamming weight of path, which diverges from state S_0 and re-merge into it for the first time at some later time. The best achievable d_H for a convolutional code with a given rate and encoder memory has not been determined exactly. However, upper and lower bounds on d_H for the best code have been obtained using a random coding approach.

2.2 Rate-Compatible Punctured Convolutional Codes

In many bandwidth-constrained application, high-rate convolutional codes are desirable. However, the implementation of Viterbi decoders, which will be discussed in later section, for these codes is often quite complicated. For the (n, k, λ) convolutional codes, the complexity of the Viterbi algorithm is proportional to $2^{k\lambda}$, where $k\lambda$ the number of decoder state. The complexity of the decoder can be overcome in many cases by a process of puncturing, which was first introduced to convolutional coding by Cain, Clark and Geist [6]. The puncturing process can obtain simple Viterbi decoding to convolutional code.

Punctured convolutional codes are a class of convolutional code. A rate $P/(P + \delta)$ punctured convolutional code can be obtained by *periodically* puncturing a low rate $R = 1/n$, memory λ convolutional code with period P and $1 \leq \delta \leq (n-1)P$. The low rate $R = 1/n$ code is called *mother code*. For a mother code (2, 1, 2) with polynomial



(a) (b)

Figure 2.5: (a) Trellis diagram for (2, 1, 2) convolutional code encoder
(b) Trellis diagram for (3, 2, 2) punctured convolutional code encoder

generator matrix

$$G(D) = [1+D^2 \;\; 1+D+D^2] \tag{2.11}$$

with trellis diagram in Figure 2.5(a), the symbol "X" in the figure means that the bit is punctured or deleted at that position. If every third encoder output bit is punctured, the code will produce three coded bits for every two input information bits and a rate 2/3 punctured code will be obtained. The resulting code is identical to the rate 2/3 with polynomial generator matrix

$$G(D) = \begin{bmatrix} 1+D & 1+D & 1 \\ 0 & D & 1+D \end{bmatrix}, \tag{2.12}$$

and the trellis diagram for punctured convolutional code is shown in Figure 2.5 (b).

The basic procedure for constructing a high rate $R = P/(P + \delta)$ punctured code from a rate $1/n$ code can be described as encoding by the mother code followed by a puncturing device. The puncturing device punctures the encoded output symbols using a $n \times P$ *puncturing matrix* $A(\delta)$ with binary elements for $1 \leq \delta \leq (n-1)P$. The rate of the punctured code is determined by the values of P and δ. For example, the rate 2/3 code is generated from the rate 1/2 mother code using the puncturing matrix

$$A(\delta) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \tag{2.13}$$

where 0 implies puncturing. In the (2.13), the period P and δ is equal to 2 and 1 respectively, i.e., one bit is punctured for every four bits.

Rate-compatible puncturing convolutional (RCPC) codes is a subclass of punctured convolutional codes [7]. Consider an example where a rate 4/7 punctured code period $P = 4$ and $\delta = 3$ is generated from a rate 1/2 mother code given in the previous example. The puncturing matrix for the rate 4/7 punctured code is

$$A(3) = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \tag{2.14}$$

where there is one zero element in the matrix $A(3)$. For rate compatibility, the punctured matrices for the rate 4/6 and rate 4/5 codes are

$$A(2) = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \tag{2.15}$$

and

$$A(1) = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \tag{2.16}$$

respectively.

To generate a rate 4/6 punctured code with same period P from the rate 1/2 mother code, we have two zero elements in the matrix $A(2)$ of (2.15). For rate compatibility, the zero elements in $A(3)$ is retained in the matrix $A(2)$. In general, all zero elements in A(δ) are retained in the subsequent matrices A(δ-1), A(δ-2), ... This restriction implies that all the code symbols of the higher rate punctured codes are required by the lower rate codes. The matrix of A(1) in (2.16) retains the zero elements of A(2) in (2.15).

RCPC codes are suitable for applications where different levels of error protection are required within an information sequence or block. A typical information sequence or block is grouped according to their error protection requirement and different rate of RCPC codes are assigned to these groups. The least error sensitive group is protected by a high rate RCPC code while more error sensitive group is protected by a low rate RCPC code. RCPC codes are used to provide unequal error protection to embedded image coder in later chapter of this thesis.

2.3 Trellis-Coded Modulation

In the modern communication systems shown in the Figure 1.1, power and bandwidth are limited resources. The complexity of the communication systems will invariably be increased for the efficient allocation of these resources. In a power-limited environment, the desired system performance should be achieved with the smallest possible power. One solution is the use of error-correcting codes, which increase the power efficiency by adding redundant bits to the transmitted sequences for error control. However, this method requires the modulator to operate at a higher data rate and hence requires a larger bandwidth. In a bandwidth limited environment, increased efficiency in frequency utilization can be obtained by choosing higher-order modulation scheme, but a larger signal power would be needed to maintain the same signal separation and hence the same error probability.

One very successful method of reducing power requirement without increasing the requirements on bandwidth was introduced by Ungerböck [8] using *trellis-coded modulation* (TCM). TCM combines convolutional encoding with signal modulation as an unique operation. A receiver also combines demodulation and decoding in a single step to process the received signal, instead of being first demodulated and then decoded. With combined encoding and modulation, the parameter governing the performance of the transmission system is not the free Hamming distance (d_H) of the convolutional code, but rather, over the additive white Gaussian noise channel (AWGN), the *free Euclidean distance* (d_{free}) between transmitted signal sequences. The optimization of the TCM design is based on the free Euclidean distance rather than on Hamming distance

2.3.1 General Model of TCM

TCM is another kind of sliding window codes [10]. The signal x_n transmitted at discrete time n not only depends on the source symbols a_n transmitted at the same time instant, but also depends on finite number of previous λ source symbols:

$$x_n = f(a_n, a_{n-1}, a_{n-2}, \ldots, a_{n-\lambda}). \tag{2.17}$$

By defining

$$S_n = (a_{n-1}, a_{n-2}, \ldots, a_{n-\lambda}) \tag{2.18}$$

as the *state* of the encoder at time n, then (2.17) can be written in the more compact form:

$$x_n = f(a_n, S_n) \tag{2.19}$$

The function $f(\cdot, \cdot)$ generates the output signal x_n according to the input a_n and S_n. The output signal x_n not only depends on the corresponding source a_n, but also depends on the previous symbols S_n. In other words, at any time instant, the transmitted symbol is chosen from a constellation that is selected by the value of S_n. Besides, the next state is:

$$S_{n+1} = g(a_n, S_n), \tag{2.20}$$

where function $g(\cdot, \cdot)$ describes the memory part of the encoder and shows the evolution of the modulator states. The general model of TCM is illustrated at Figure 2.6.

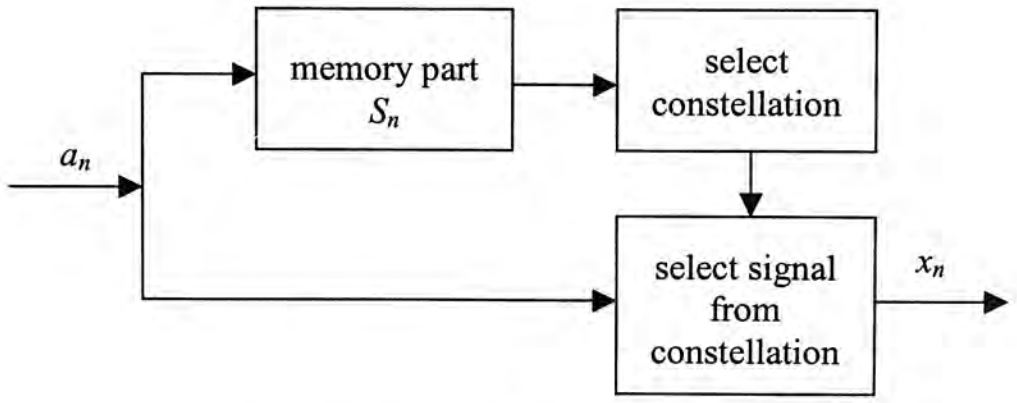


Figure 2.6: General model for TCM

2.3.2 Trellis Representation

Functions $f(\cdot,\cdot)$ and $g(\cdot,\cdot)$ in (2.19) and (2.20) can be represented by *trellis*. The values S_n, the encoder state at time n, are the nodes of the trellis. With each source symbol, a branch is stemmed from each modulator state at time n and reaches the next encoder state at time n+1. The branch is labeled by the notation ***a*/*b***, where ***a*** corresponds to the input source symbol a_n and ***b*** corresponds to the output value of $f(\cdot,\cdot)$. The trellis structure is determined by the function $g(\cdot,\cdot)$, while $f(\cdot,\cdot)$ describes how channel symbols are associated with each branch along the trellis. For M-ary source symbols, each node must have M branches stemming from it (one per each source symbol), implying that two or more branches may connect the same pair of nodes, namely *parallel transition* between nodes.

Figure 2.7 shows an example of trellis representation, assuming that the encoder has four states, the source emits binary symbols 0 and 1, and a constellation with four signals x_0, x_1, x_2, x_3, corresponding to 0, 1, 2, 3 in Figure 2.7, respectively.

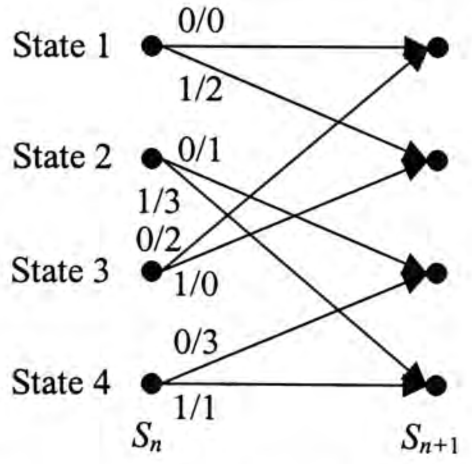


Figure 2.7: Trellis representation for TCM with four states and four channel symbols used to transmit from a binary source

2.3.3 Set Partitioning

Using the trellis diagram, the minimum distance d_{free} of TCM is the distance between the signal associated with the pair of paths that originate from an initial split and, after T time instants, merge into a single node [10], as shown in Figure 2.8. If the free distance is determined by parallel transitions (i.e., T = 1), d_{free} equals the minimum distance among signals in emanating from a give node.

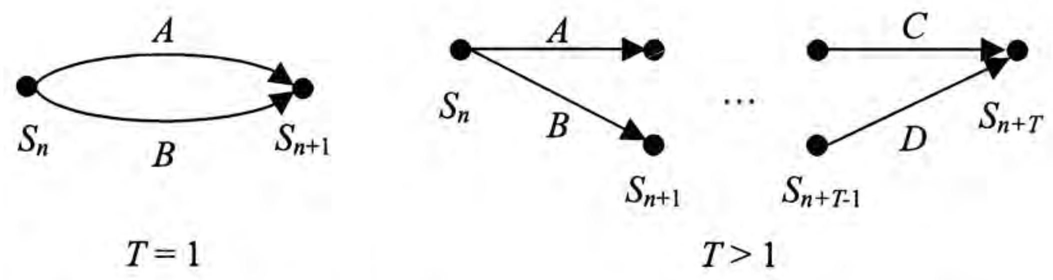


Figure 2.8: Pair of splitting and remerging paths for $T = 1$ (parallel transitions) and $T > 1$

Consider next T>1, using A, B, C, D to denote subsets of signals associated with each branch, and $d(X, Y)$ denoting the minimum Euclidean distance between one signal in X and one in Y, d_{free} has the expression

$$d_{free}^2 = d^2(A,B) + \cdots + d^2(C,D) . \tag{2.21}$$

To obtain a good code, the subsets assigned to the same originating state or to the same terminating state must have the largest possible distance.

To implement these rules, Ungerböck [8] develops a scheme called *set partitioning*. The key point in partitioning of constellation is to find subsets of the constellation that are similar and the points inside each partition are maximally separated. Starting from the original constellation, we partition the constellation into two subsets that are congruent and the points within each partition are separated maximally. The same principle can be applied to each partition separately. The rules for set partitioning can be summarized as following:

1. Parallel transitions (when they occur) correspond to signal points in a single partition at the next stage of partitioning
2. The transitions originating from and merging into any state are assigned partitions in the next stage of partitioning that have a single parent partition in the preceding stage.
3. The signal points should occur with equal frequency.

The above three rules are referred to hereafter as the three *Ungerböck rules*. Figure 2.8 and Figure 2.9 show the examples of set partitioning for 8-AM and 8-PSK constellation respectively. These examples will be used in later chapters of this thesis.

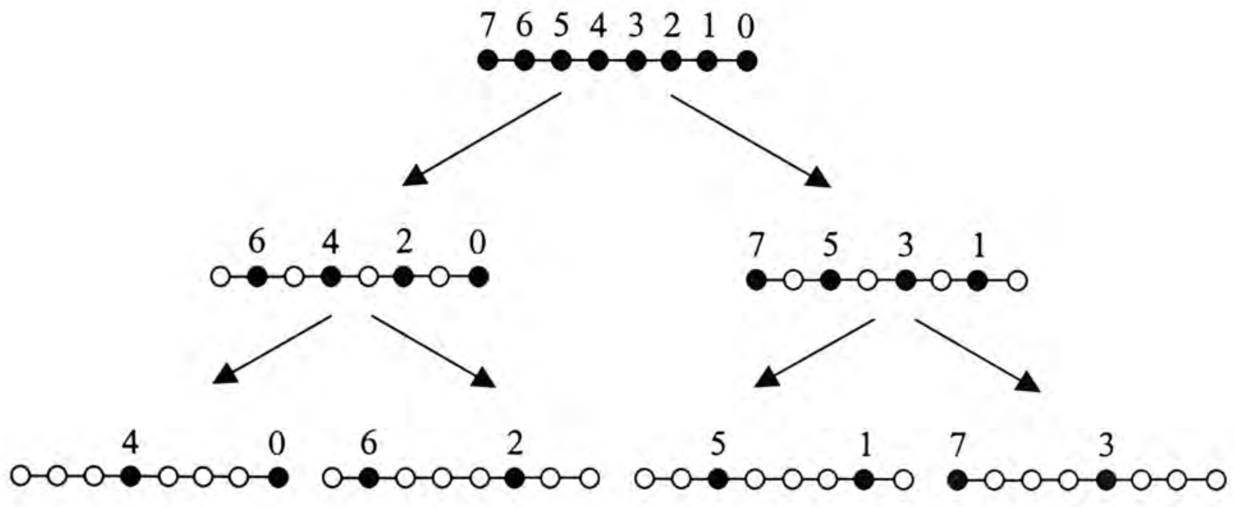


Figure 2.8: Set partition for 8-AM

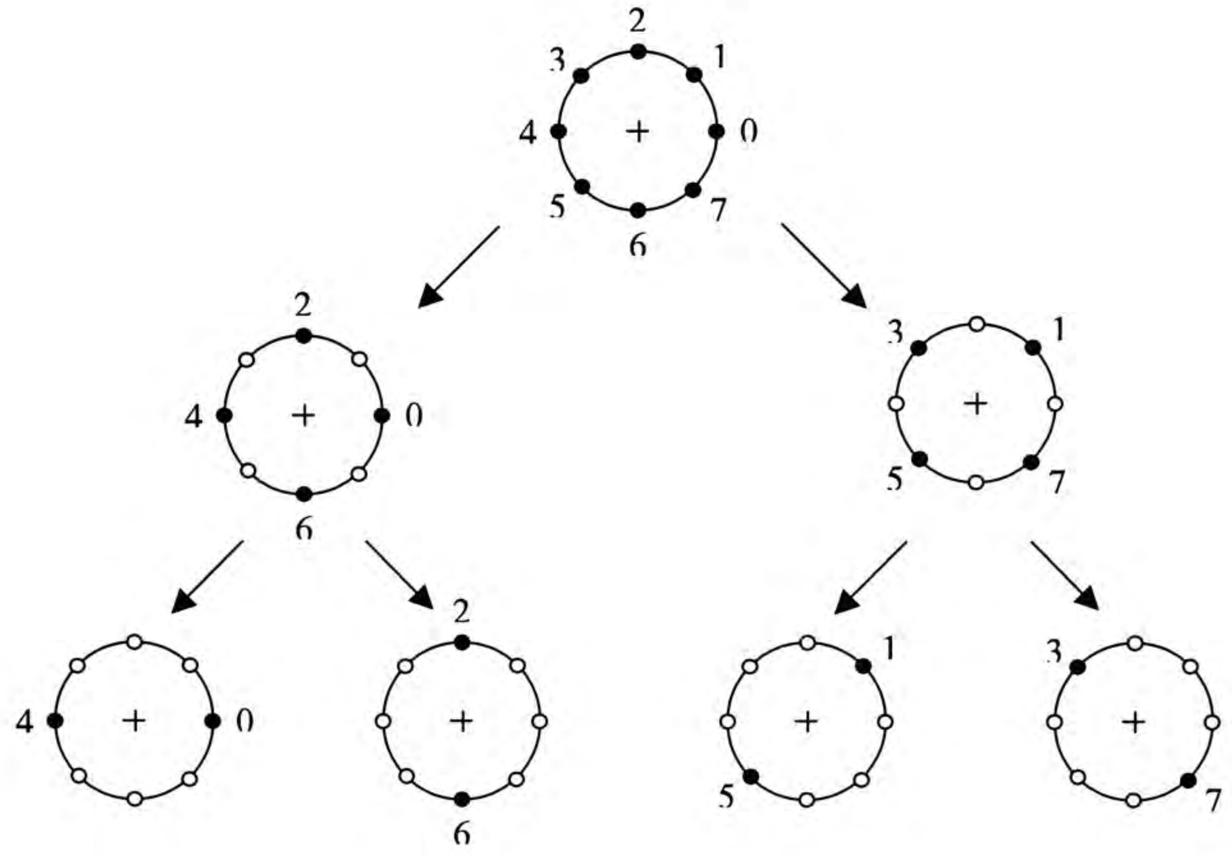


Figure 2.9: Set partition for 8-PSK

2.3.4 Coded Modulation

The block diagram of a coded modulation scheme is shown in Figure 2.10. A block of length k input bits is divided into two sub-blocks of length k_1 and k_2 respectively. The first k_1 bits are applied to a (k_1, n_1) binary encoder. The output of the encoder consists of n_1 bits. These bits are used to choose one of 2^{n_1} partitions in the constellation. In other words, the constellation has been partitioned into 2^{n_1} subsets. After the constellation is chosen, the remaining uncoded k_2 bits are used to choose one of points in the chosen constellation, having 2^{k_2} points in each partition. Therefore, the partitioning that is used contains 2^{n_1} subsets and each subset

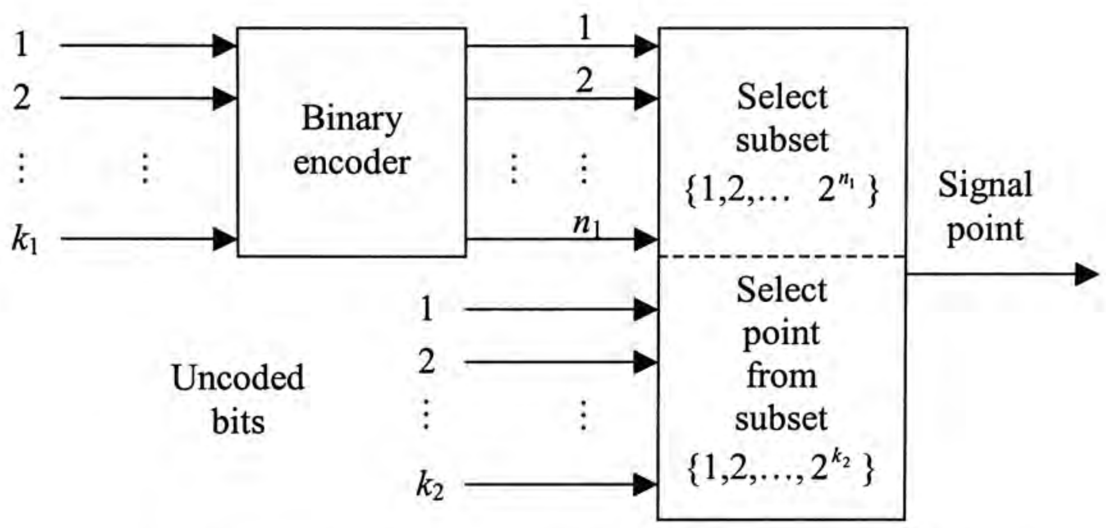

1
2
⋮
k_1
Binary encoder
1
2
⋮
n_1
Select subset $\{1,2,\ldots\ 2^{n_1}\}$
Signal point
Uncoded bits
1
2
⋮
k_2
Select point from subset $\{1,2,\ldots,2^{k_2}\}$

Figure 2.10: The block diagram of a coded modulation system

contains 2^{k_2} points. The presence of uncoded bits causes parallel transitions for TCM.

In 1982, the Ungerböck proposed a coding scheme called Ungerböck codes, which is very important in the family of TCM codes. Ungerböck chooses $n_1 = k_1+1$ and $k_2 = k-n_1$ and using convolutional codes with code rate of $k_1/(k_1+1)$ as binary encoder. Ungerböck codes can achieve overall coding gain between 3 and 6 dB [9].

In Figure 2.10, the binary encoder for the simple codes can be specified by the *parity check matrix*. For example, the parity check matrix for 4-AM, four-state trellis code is

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{2.22}$$

From the parity check matrix, it is easy to construct the block diagram. "1" in the both ends of first row means feedback. "0" means no connection between them. The data source is shifted into the encoder in the row direction. Figure 2.11 shows the

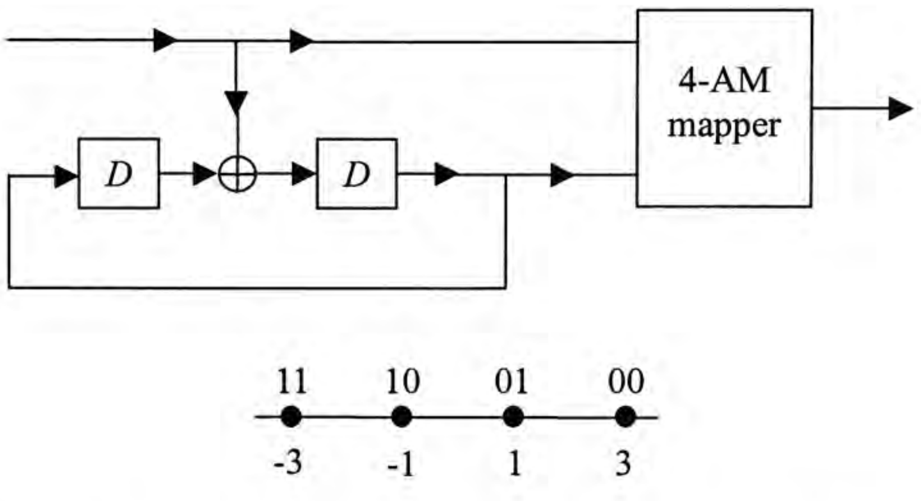


Figure 2.11: Encoder for 4-AM, four-state trellis code with natural mapper

feedback form of 4-AM, four-state trellis code with natural mapper. For all TCM schemes, circuits without feedback could be used as the encoder instead [10].

2.4 Decoding Algorithm

Previous sections show that convolutional code and trellis coded modulation can be described by using a trellis, whose branches are associated with transition between encoder states and with signal transmitted over the channel. The task of the decoder is to estimate the path that the encoded signal sequence traverses through the trellis. The decoding can be done by associating each branch of the trellis with a number, called the *branch metric*, and looking for the path whose total metric is minimum. The path with minimum total metric corresponds to the best estimation of the transmitted signal sequence.

In the decoding schemes, there exists the possibility of *soft-decision decoding* and *hard-decision decoding*. In soft-decision decoding, assume that the received signal sequence is $\mathbf{r} = \{r_0, r_1, \ldots, r_{k-1}\}$ where r_i is M-ary symbols, and assume x_j represents one of signal points in the constellation. The received signal sequence **r** is compared to the various sequences of signal points in the constellation of the coded modulation system and the one, which is closest to it in *Euclidean distance*, is chosen. The branch metric in the soft-decision decoding is defined as:

$$\|r_i - x_j\|^2 \tag{2.23}$$

In hard-decision decoding, the received signal vector **r** is first quantized into a binary sequence $\mathbf{y} = \{y_0, y_1, \ldots, y_{k-1}\}$ by making decisions on individual components of **r**, and then the codeword, which is closest to **y** in the *Hamming distance*, is chosen. The branch metric in hard-decision decoding is defined as:

$$d(y_i - x_j) \tag{2.24}$$

The fundamental task in above two approaches is to find a path through the trellis that is at minimum distance from a given sequence. The soft-decision decoding can provide up to 3 dB additional gains over the hard-decision decoding by eliminating the quantization error associated with hard-decision decoding. The same coding performance in hard-decision decoding can be achieved by soft-decision decoding using half the transmission power and resulting with cost reduction in circuit and bandwidth.

In the following sections, we introduce a practical maximum-likelihood decoding algorithm, *Viterbi algorithm*, which can be used to decode both convolutional code and trellis-code modulation.

2.4.1 Viterbi Algorithm

Viterbi algorithm can be used to decode convolutional codes and TCM, which are based on the trellis diagram representing the encoder. Viterbi algorithm finds the path with minimum path metric by sequentially moving through the trellis stage by stage. In every stage, the receiver receives the sample r_i and computes the metric associated with all branches. The path metric for a particular path through a given node is the sum of the partial path metrics for the portion of the path prior the node and the portion after the node. Among the possible paths prior to the node, the decoder prefers the one with the smallest partial path metric, called the *survivor path* for that node. All partial paths prior to the node other than the survivor path are removed from consideration. Thus, for each node, the algorithm only stores the survive path and its partial metric to simplify the complexity of the algorithm.

At each stage of the trellis, the algorithm must keep one survivor path for each and every state because the state, which the *optimal path* (or *best path*) passes through, is not known before hand. To get this survivor path for a given state at stage k, the algorithm looks at all the branches leading to that state from state k-1, and at the partial path metric of the states of those branches. The algorithm determines the partial path metric for each of those paths by summing the partial path metric of the survivor at time k-1 and the branch metric. The survivor path at that state is chosen as the path with the smallest path metric.

The algorithm stores, for each state at stage k, the survivor path and the associated path metric. The algorithm proceeds to time $k+1$ by computing the new branch metrics, and so on.

In this thesis, our presentation for Viterbi algorithm follows [11]. It finds the best state sequence through the trellis. The algorithm is summarized as following:

Assume that the there are total N states each trellis stage and total number of stage is M. The branch metric associated with moving from state j at time t-1 to state i at time is given by $C_t(j, i)$ where $C_t(j, i) = \infty$ if j and i are not connected. Now, the algorithm are going to find the best state sequence having the minimum metric through the trellis, starting from, for simplification, state 1 (at time 0) and ending at state 1 (at time M).

Let the minimum cost to reach state j at time t from the known starting state 1 be $W_t(j)$ and the survivor path be stored in the array $\Pi_t(j)$. At time t, $\Pi_t(j)$ is the state occupied by the survivor path into state j at time t-1. It is shown in the Figure 2.12.

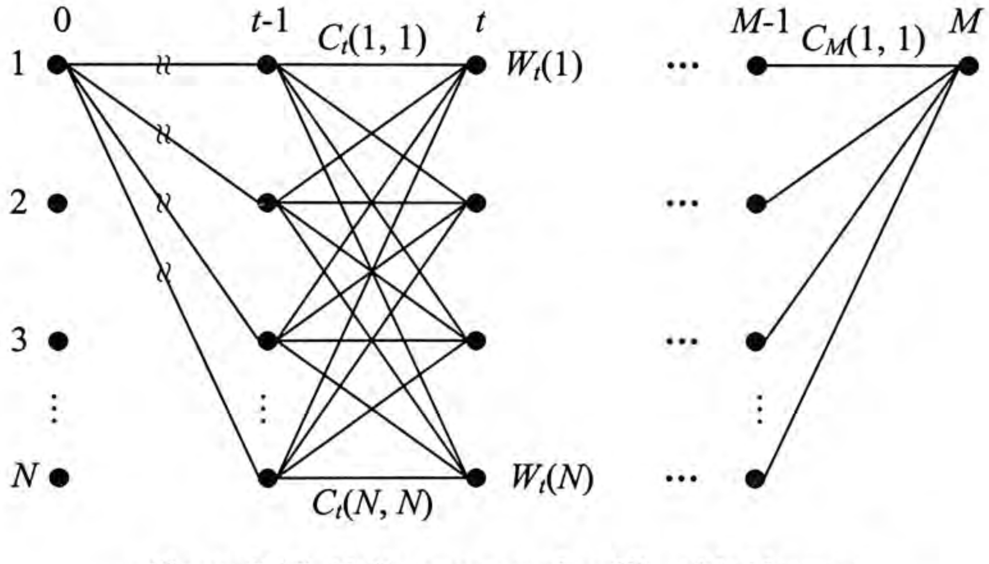


Figure 2.12: Fully connected trellis with N states

The decoding process operates as following:

1. Initialization: (t=1)

   $W_1(i) = C_i(1, i)$,

   $\Pi_1(i) = 1$,

   $1 \leq i \leq N$

2. Recursion: ($1 < t < M$)

   $W_t(i) = \min\limits_{1 \leq j \leq N}[W_{t-1}(j) + C_t(j,i)]$,

   $\Pi_t(i) = \arg\min\limits_{1 \leq j \leq N}[W_{t-1}(j) + C_t(j,i)]$,

   $1 \leq i \leq N$

3. Termination: ($t = M$)

   $W_M(1) = \min\limits_{1 \leq j \leq N}[W_{M-1}(j) + C_M(j,1)]$,

   $\Pi_M(1) = \arg\min\limits_{1 \leq j \leq N}[W_{M-1}(j) + C_M(j,1)]$,

4. Path Backtracking:

   The best state sequence is:

   $(1, i_1, \ldots, i_{M-1}, 1)$,

   where

   $i_t = \Pi_{t+1}(i_{t+1})$

   $1 \leq t \leq M-1$

The best state sequence is: $(1, i_1, \ldots, i_{M-1}, 1)$

Viterbi algorithm never declares an uncorrectable error pattern and always produces a best (maximum-likelihood) path as decoder output.

2.4.2 List Viterbi Algorithm

In the previous Viterbi algorithm, it only finds the best path traversing the trellis. A more general list Viterbi algorithm (LVA) [11] provides a rank ordered list of the L globally best path after a trellis search. Viterbi algorithm can be treated as a special case of list Viterbi algorithm when $L = 1$.

The algorithm is illustrated in Figure 2.13, at each state and at every time, the NL accumulated costs are computed and the L smallest accumulated cost paths along with their costs are stored. This algorithm requires maintaining a cost array of NL accumulated costs and a state array of $NL \times M$, which stores the path history for each time instant.

The algorithm is summarized as following:

Let $W_t(i, k)$, $1 \leq k \leq 1$, be the k^{th} lowest cost to reach state i at time t from state 1 at time 0. Similarly let $\Pi_{t-1}(i, k)$ be the state of the k^{th} best path at time t-1, when

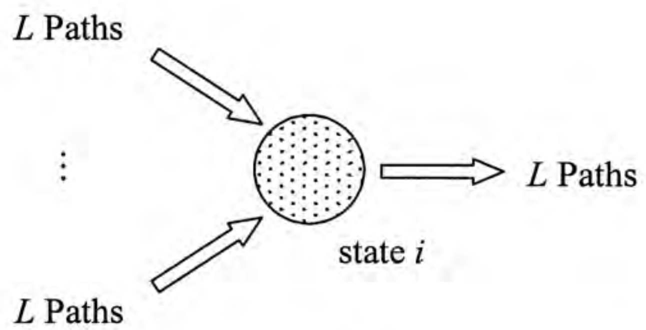


Figure 2.13: A state for parallel list Viterbi algorithm

this path passes through state i at time t

So the decoding process operates as following:

1. Initialization: (t=1)

$W_1(i, k) = C_i(1, i)$,

$\Pi_1(i, k) = 1$,

$1 \le i \le N, 1 \le k \le L$

2. Recursion: ($1 < t < M$)

$$W_t(i, k) = \min_{\substack{1 \le j \le N \\ 1 \le l \le L}}{}^{(k)}[W_{t-1}(j,l) + C_t(j,i)], \quad 1 \le i \le N$$

where $\min^{(k)}$ denotes the k^{th} smallest value

$$(j^*, l^*) = \arg\min_{\substack{1 \le j \le N \\ 1 \le l \le L}}{}^{(k)}[W_{t-1}(j,l) + C_t(j,i)], \quad 1 \le i \le N$$

Here j^* is the predecessor state for the k^{th} best path into state i and l^* is the corresponding ranking.

$W_t(i, k) = j^*$,

$r_t(i, k) = l^*$.

3. Termination: ($t = M$)

$$W_M(1, k) = \min_{\substack{1 \le j \le N \\ 1 \le l \le L}}{}^{(k)}[W_{M-1}(j,l) + C_M(j,i)], \quad 1 \le i \le N$$

$$(j^*, l^*) = \arg \min_{\substack{1 \le j \le N \\ 1 \le l \le L}}{}^{(k)}[W_{M-1}(j,l) + C_M(j,i)], \quad 1 \le i \le N$$

$W_t(i, k) = j^*$,

$r_t(i, k) = l^*$.

4. Path Backtracking:

The best state sequence is:

$(1, j_1, \ldots, j_{M-1}, 1)$,

where

$j_t = \Pi_{t+1}(j_{t+1}, l_{t+1})$,

$l_t = r_{t+1}(j_{t+1}, l_{t+1})$,

$j_{M-1} = \Pi_M(1, k)$,

and

$l_{M-1} = r_M(1, k)$,

This algorithm can produce L best path in one pass through the trellis, so it is also called *parallel list Viterbi algorithm* (PLVA)

Chapter 3

Unequal-Error-Protection for Embedded Image Coder

When discrete wavelet transform and set-partitioning-in-hierarchical-trees (SPIHT) [12] are combined together, in additional to good compression ratio, image is compressed into many layers and transmission can be stopped in any data-rate. SPIHT is also a low-complexity fully embedded wavelet-coding algorithm that can facilitate precise rate control with progressive transmission. The more bits are transmitted, the better quality image can be reconstructed at the receiver. Both transmitter and receiver can stop transmission or decoding at any time to provide the best image up to the stopped data-rate. In SPIHT, high layer data are always more important than lower layer data.

Transmitted through a communication system, due to the unavoidable noise, SPIHT compressed data may subject to random channel bit error. SPIHT algorithm is very sensitive to channel error. A single error in the SPIHT compressed data results in non-recoverable loss of synchronization between the encoder and decoder until the next re-synchronization point. The process of image decompression may require to pause at the first bit-error because of the dependence of lower layers on previous transmitted layers. Therefore, the higher layers are not only more important because of its better resolution, but also the dependence of lower layers on higher layers.

Most of the previous studies of progressive image transmission used equal error protection on the whole image [13]. Different rate-compatible punctured convolutional (RCPC) codes [7] are used for channel having different bit error rate (BER) without taking the advantage that different layers have different importance, and thus may subject to different error protection. In equal error protection scheme, all high layer data may have insufficient error protection but lower layer data may have too much error protection.

Optimal channel code allocation was studied in [14] to provide unequal error protection but a complicated algorithm was proposed that required real-time implementation instead of off-line calculation. The complexity of the algorithm in [14] is in the order of $O(N^2)$. Hence, we are going to derive a simple and low complexity algorithm for channel code allocation to provide unequal error protection for SPIHT coded image transmitted over binary symmetric channels (BSC) [15]. RCPC codes with different punctured rate are used to provide unequal error protection for each data packet to minimize the end-to-end image distortion. The code allocation algorithm can achieve better peak signal to noise ratio (PSNR) at a wide range transmission rate and the algorithm has a complexity of only *O*(*N*), which is far smaller than that of [14].

3.1 SPIHT Coder

The SPIHT image coding algorithm was developed in 1996 by Said and Pearlman [12] and is an efficient implementation of the embedded zerotree wavelet (EZW) algorithm by Shapiro [16]. The EZW technique is based on the following three concepts:

1. Partial ordering of the transformed image elements by magnitude, with transmission order by a subset partitioning algorithm
2. Ordered bit plane transmission of refinement bits, and
3. Exploitation of the self-similarity of image wavelet transform across different scales

The partial ordering is a result of comparing transform coefficient magnitudes to a set of octavely decreasing thresholds. An element is significant or insignificant with respect to a given threshold, depending on whether or not it exceeds that threshold.

The crucial part of coding processing for SPIHT is that the way of subsets of coefficient is partitioned and how the significance information is conveyed which is fundamentally different from EZW.

3.1.1 Progressive Image Transmission

Assume that the original image is defined by a set of pixel values $p_{i,j}$, where (i, j) is the pixel coordination. The coding is conducted to the array

$$\mathbf{c} = \Omega(\mathbf{p}), \tag{3.1}$$

where $\Omega(.)$ represents a unitary hierarchical subband transformation. The 2-D array $\mathbf{c}$ has the same dimensions of $\mathbf{p}$, and each element $c_{i,j}$ is called *transform coefficient* at coordinate (i, j).

In the progressive transmission scheme, the decoder initially sets the reconstruction vector $\hat{c}$ to zero and updates its components according to the coded message. After receiving the of some coefficients, the decoder can obtain a reconstructed image

$$\hat{p} = \Omega^{-1}(\hat{c}) . \tag{3.2}$$

A major objective in a progressive transmission scheme is to select the most important information, which yields the largest distortion reduction, to be transmitted first. For this selection, *mean squared-error* (MSE) is used to measure distortion

$$D_{MSE}(p - \hat{p}) = \frac{\|p - \hat{p}\|^2}{N} = \frac{1}{N}\sum_i \sum_j (p_{i,j} - \hat{p}_{i,j})^2 , \tag{3.3}$$

where N is total number of image pixels. Furthermore, a fact that the Euclidean norm is invariant to the unitary transform Ω, i.e.

$$D_{MSE}(p - \hat{p}) = D_{MSE}(c - \hat{c}) = \frac{1}{N}\sum_i \sum_j (c_{i,j} - \hat{c}_{i,j})^2 . \tag{3.4}$$

If the exact value of the transform coefficient $c_{i,j}$ is sent to the decoder, MSE decreases by $\left|c_{i,j}\right|^2 / N$. Therefore, the coefficients with larger magnitude should be transmitted first because they contain more information of image. The maximum number of bits required to represent the largest coefficient is denoted as $n_{\max}$, which is

$$n_{\max} = \left\lceil \log_2\left(\max_{(i,j)} \left\{ \left| c_{i,j} \right| \right\} \right) \right\rceil, \tag{3.5}$$

where symbol $\lceil \ \rceil$ means rounding up the element inside it to nearest integer. The bit can be controlled precisely in the SPIHT algorithm because the outputs are in single bits and the algorithm can be terminated at anytime. The decoding process follows the encoding process exactly and is almost symmetrical in terms of processing time.

3.1.2 Set Partitioning Sorting Algorithm

The main algorithm partitions the wavelet decomposed image into significant and insignificant partitions based upon the following function

$$S_n(T) = \begin{cases} 1, & \max_{(i,j)\in T} \left\{ \left| c_{i,j} \right| \right\} \geq 2^n, \\ 0, & \text{otherwise,} \end{cases} \tag{3.6}$$

where $S_n(T)$ is the significant of a set of coordinates T, and $c_{i,j}$ is the coefficient value at coordinate (i, j). There are two passes to the algorithm, the *sorting pass* and the *refinement pass*. The sorting pass is performed on the list of insignificant set (LIS), list of insignificant pixels (LIP) and the list of significant pixels (LSP). The LIP and

LSP consists of nodes that contain single pixels while the LIS contains nodes that have descendants.

3.1.3 Spatial Orientation Trees

In general, most of an image's energy is concentrated in the low frequency components. The variance decreases as moving from the highest to the lowest levels of the subband pyramid. Also it has been observed that there is a spatial self-similarity between subbands, and the coefficients are expected to be better magnitude-ordered if we move downward in the pyramid following the same spatial orientation. For example, large low-activity areas are expected to be identified in the highest levels of the pyramid, and they are replicated in the lower levels at the same spatial.

A tree structure, call *spatial orientation tree*, describes the relationship on the hierarchical pyramid. Figure 3.1 shows how the spatial orientation tree is constructed in pyramid with recursive four-subband splitting. Each node of the tree corresponds to a pixel and identified by the pixel coordinate. The direct descendants (offspring) of each node correspond to the pixels of the same spatial orientation in the next finer level of the pyramid. The tree is defined in such a way that each node has either no offspring (the leaves) or four offspring, which always form a group of 2×2 adjacent pixels. In Figure 3.1, the arrows are oriented from the parent node to its four offspring. The pixels in the highest level of the pyramid are the tree roots and are also grouped in 2×2 adjacent pixels. However, their offspring branching rule is

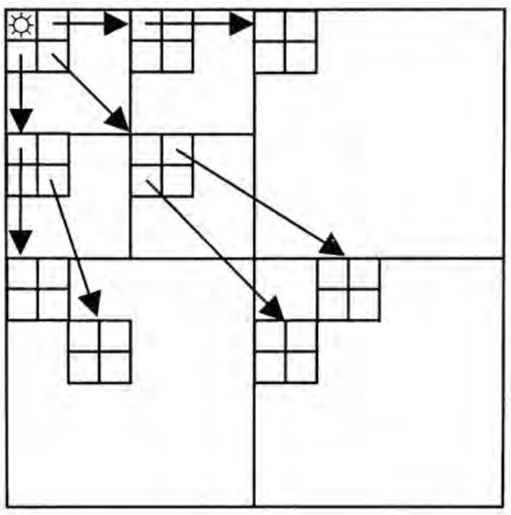


Figure 3.1: Example of parent-offspring dependencies in the spatial-orientation tree

different, and in each group, one of them (indicated by star) has no descendants.

3.2 System Description

In our system, a SPIHT coded image is transmitted over the binary symmetric channel (BSC). Like the method of [13], a concatenated coding system is used which consists of an outer error detecting code and an inner error correcting code to protect the SPIHT coded image.

The whole image is compressed using the SPIHT compression algorithm without arithmetic coding [12]. The compressed image is then put into different packets with 200 bits each. Each packet is passed to a concatenated encoder consisting of 16-bit cyclic redundancy code (CRC) checksum bits outer code followed by a RCPC inner code, the same as that in [13]. The binary RCPC code with memory of six is used as error correction code.

At the receiver side, list Viterbi algorithm (LVA) in [11] is used to decode the RCPC inner code. The path with lowest metrics and satisfying CRC checksum is chosen as decoded output. The maximum number of searched is limited to 100. After searching first 100 best paths, if none of them satisfies the CRC checksum, the decoder treat it as decoding error and stop further decoding. The lower layers do not improve image quality due to error dependence of SPIHT decoder. The including of lower layers may even degrade the quality of the decoded image. The concatenated coding system is shown in Figure 3.2. The LVA could be implemented in parallel to evaluate all possible paths together, especially in hardware implementation.

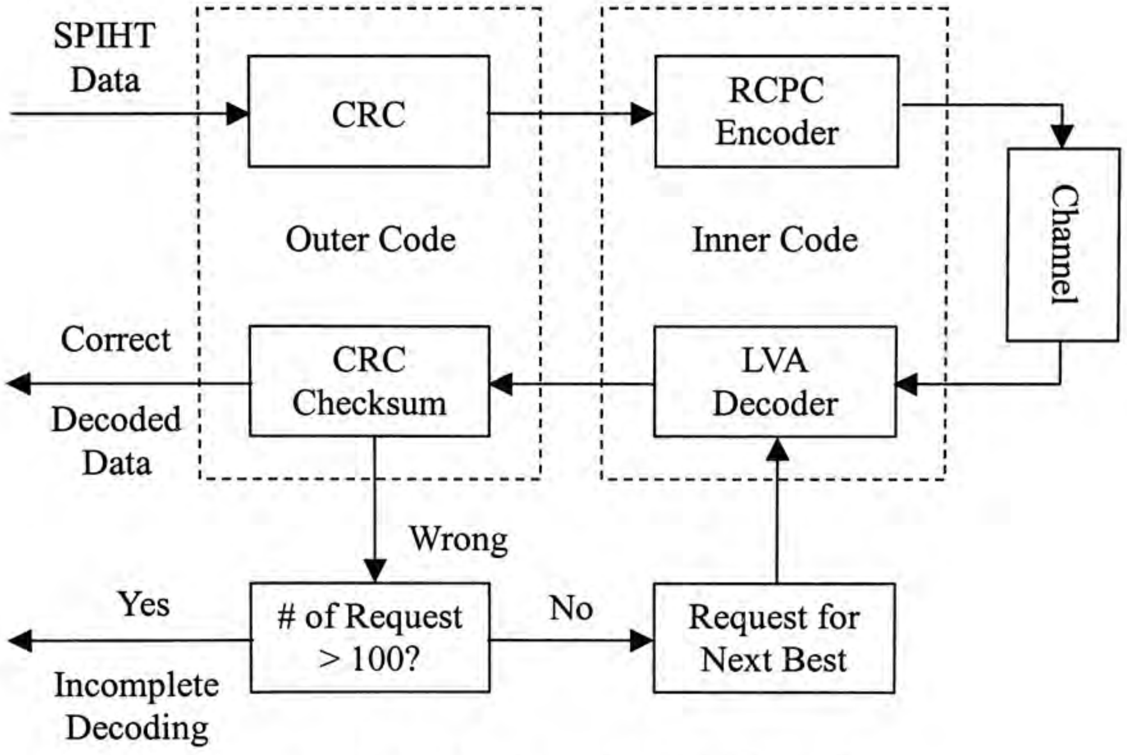


Figure 3.2 Concatenated coding system

3.3 Code Allocation

Unequal error protection can be provided by RCPC [7]. RCPC codes can use the same decoder structure or hardware to decode a set of codes with different error protection capabilities. Different channel code rates can be implemented through changing the encoder and decoder just by varying the number of punctured bits for RCPC codes. The same decoding software or hardware can be used by "ignoring" the punctured position of the RCPC.

Assuming that the distortion with a total of k packets transmitted is D_k, i.e., D_0, D_1, D_2, ... in decreasing order, where D_0 is the total energy of the image. If a total number of N packets are transmitted through a noisy channel with unequal error protection, the expected distortion of the image is

$$D_T = \sum_{k=1}^{N} p_k \prod_{j=1}^{k-1} (1 - p_j) D_{k-1} , \tag{3.7}$$

where p_k are the error probability of the kth packet. The above expression assumes that if an error is detected in a packet, all bits of that packet and later packets are discarded. Although special SPIHT decoder may be used to utilize the later correct packets, the discarding of coming packets is the simplest and straightforward way in most implementation. In general, $\prod_{j=1}^{k-1}(1-p_j)$ is approximately equal to unity and

$$D_T \approx \sum_{k=1}^{N} p_k D_{k-1} . \tag{3.8}$$

In (3.8), the overall distortion is approximately equal to the summation of the error probability and the distortion of each packet. This linear relationship can be very useful to simplify the code allocation problem. Usually, the user would like to preserve the quality of the image up to certain level, i.e, $D_T < D_c$ or $\sum_{k=1}^{N} p_k D_{k-1} < D_c$. The required error rate of each packet is

$$p_k < \frac{D_c}{ND_{k-1}}. \tag{3.9}$$

As D_c/N is a constant, it can be replaced by a new constant C called *adaptive constant*. The above equation can further be simplified to

$$p_k < \frac{C}{D_{k-1}}. \tag{3.10}$$

The channel code rate of each packet can be determined by the above expression. In the simplification from (3.9) to (3.10), the adaptive constant of C losses its original physical meaning of D_c/N, i.e., the average allowable distortion per packet.

Instead of using a fixed punctured rate in [13] for uniform error protection, unequal error protection is used for different packet according to (3.9) or (3.10). Once the channel bit error rate (BER) is given, the RCPC code of each packet can be decided according to (3.9) or (3.10)

3.4 System Complexity

The only required information of the algorithm from the image coder is the list of distortions of D_k, the transmitter only performs one division operation to determine the code rate for next packet according to equation (3.10). Therefore, our algorithm has a complexity of $O(N)$ instead of $O(N^2)$ in [14]. The algorithm is simple to implement.

In a conventional straightforward way, the distortions of D_k are evaluated by first encode and then decode the image with specific data-rate. In practice, the distortions of D_k can be approximately evaluated using the threshold level of the SPIHT algorithm [12]. Because the discrete wavelet transform and inverse discrete wavelet transform is a pair of unitary transform as shown in (3.4), the distortions can be evaluated using the wavelet transformed coefficients without decoding the SPIHT-coded image.

3.5 Simulation Result

The simulation used standard 512 × 512 gray scale Lena image as shown in Figure 3.3. The image was compressed by SPIHT without using arithmetic coding. The compressed image was encoded by 16-bit CRC and then followed by a RCPC code. Finally, the output of binary data was transmitted through a BSC with a BER of 0.1 and transmission rates are range from 0 bpp up to 1 bpp, including the redundancy bits for error protection.

In [13], the selected CRC polynomial is $X^{16} + X^{14} + X^{12} + X^{11} + X^8 + X^5 + X^4 + X^2 + 1$ which is used to check each decoded trellis path. The RCPC has punctured rate of 2/7 and memory of 6 whose mother code rate is a 1/4-rate convolution code having a memory of 6 and generation matrix of

Figure 3.3: Standard 512 × 512 gray scale Lena image

$$G(D) = \begin{bmatrix} 1 + D^2 + D^3 + D^4 + D^6 \\ 1 + D^2 + D^3 + D^4 + D^6 \\ 1 + D + D^4 + D^5 + D^6 \\ 1 + D + D^2 + D^5 + D^6 \end{bmatrix}^T \tag{3.11}$$

The RCPC code sets have rates of {2/7, 4/13, 1/3, 4/11, 4/10} as shown in Table 3.1. The puncturing position is also shown in Table 3.1. The zeros in the puncturing position matrix mean that it is punctured.

Different code rate has different error protection capability. Thousand of simulations were conducted for each code to obtain their decoding error probability over BSC with BER = 0.1. Table 3.1 also shows the simulation result of decoding error probability of difference RCPC codes.

Code No	Code Rate	Puncturing Position	Decoding Error Probability
1	2/7	$\begin{bmatrix} 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&0&1&0&1&0&1&0 \end{bmatrix}$	≈ 0.0
2	4/13	$\begin{bmatrix} 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&0&0&0&1&0&0&0 \end{bmatrix}$	1.0×10^{-6}
3	1/3	$\begin{bmatrix} 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}$	2.45×10^{-4}
4	4/11	$\begin{bmatrix} 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&1&0&1&1&1&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}$	2.715×10^{-3}
5	2/5	$\begin{bmatrix} 1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1 \\ 1&1&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}$	6.328×10^{-2}

Table 3.1: Decoding error probability for each packet over BSC with BER=0.1

The relation between image distortion and number of packet transmitted is shown in Figure 3.4. The more packets are transmitted, the less image distortion is obtained. The allowable error probability of each packet can be determined based on (3.10). The allowable error probability is shown in Figure 3.5. From Figure 3.5, the allowable error probability of each packet increases with packet number.

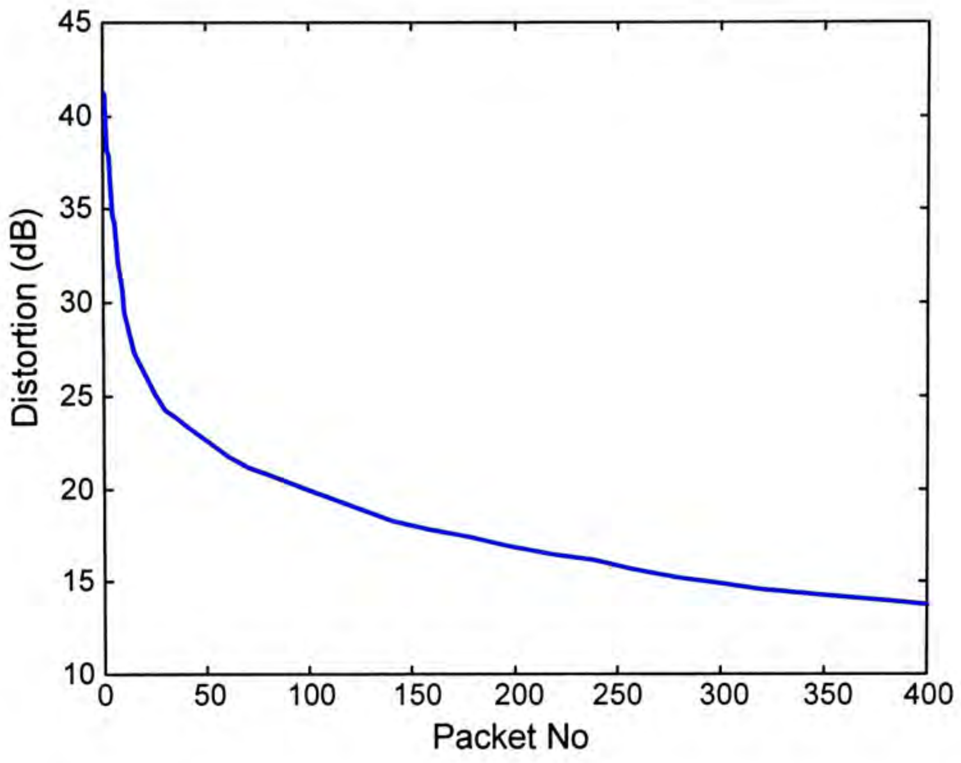


Figure 3.4: Total image distortion when *k* packets are received

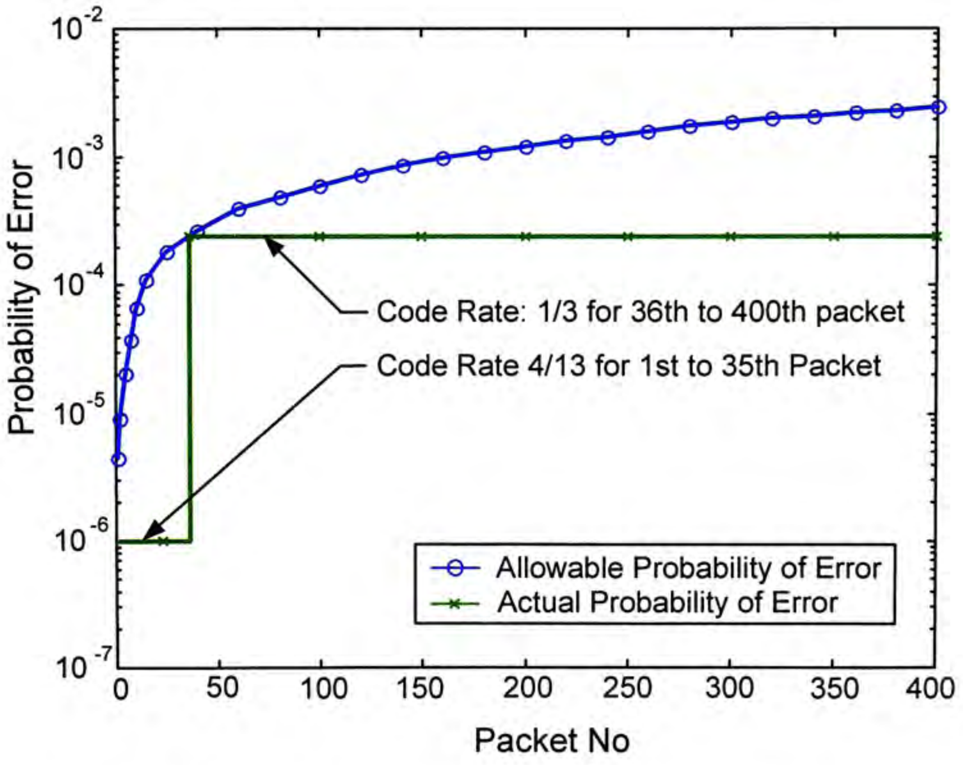


Figure 3.5: Allowable and actual probability of error for each packet

Using packet error probability and Table 3.1, the code rate for a particular packet can be selected. The actual error probability of each packet is also shown in Figure 3.5. To cite an example, $D_{35} = 242.3$ and $C = 0.06$, from (3.10), we get the allowable $p_{36} < 2.48\text{x}10^{-4}$. Because p_{36} is just larger than $2.45\ \text{x}10^{-4}$ in fourth row of Table 3.1, the code rate for 36th packet can be found to be 1/3. To preserve the quality of the image up to certain level, the actual error probability should be smaller than the allowable error probability of packet as from (3.10). Based on this scheme, we can determine the code rate for each packet. The code rate 4/13 is assigned to the 1st to 35th packet, and the code rate 1/3 is assigned to the 36th to 400th packet. As the error probability of the code rate 2/7 almost tends to zero, the code rate is immediately switched to the code rate 4/13 for the first packet.

After conducting many trials of simulation, better peak-signal-to-noise-ratio (PSNR) performance is obtained. The PSNR is defined as

$$\text{PSNR} = 10\log_{10}\left(\frac{255^2}{D_{MSE}(p-\hat{p})}\right)dB, \tag{3.11}$$

when adaptive constant C is equal to 0.06 for wide range of bit rate from 0 to 1 bpp. The non-adaptive curve in the Figure 3.6 is the result of [13] using a uniform code rate of 2/7. Our proposed algorithm can yield 0.4 to 0.8 dB performance improvements in PSNR over the equal error protection scheme of [13].

In the Figure 3.7, it shows the decoded Lena image at total bit rate 1.0 at receiver side after passing through channel with BER = 0.1. Comparing Figure 3.3 with Figure 3.7, there is almost no large difference with the original image.

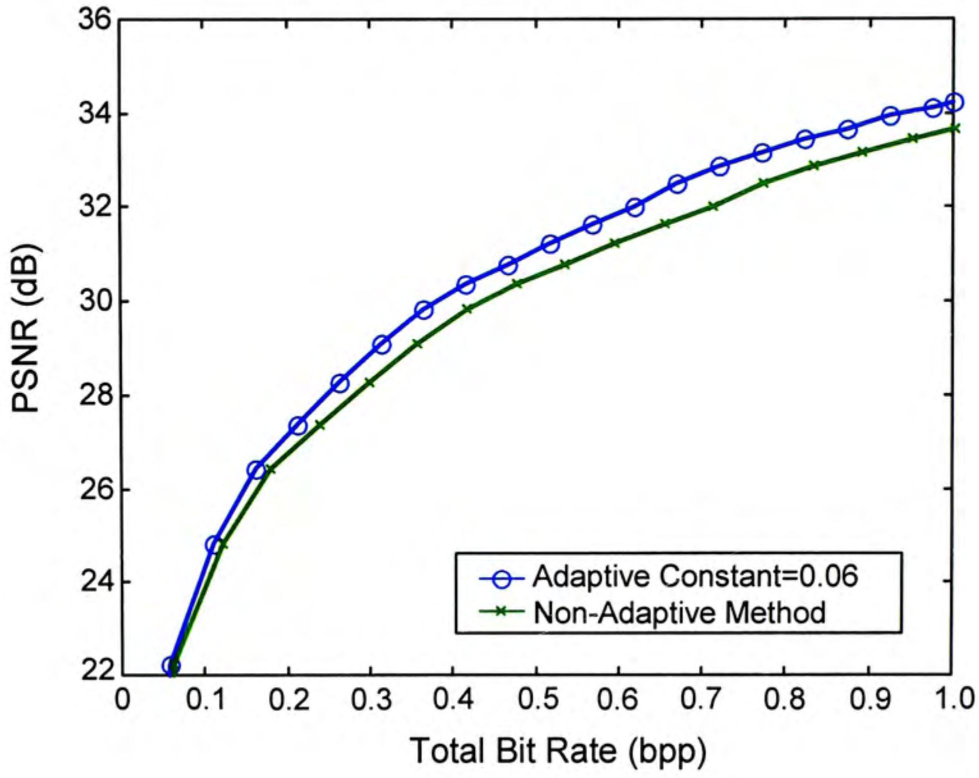


Figure 3.6: Lena over BSC with BER=0.1

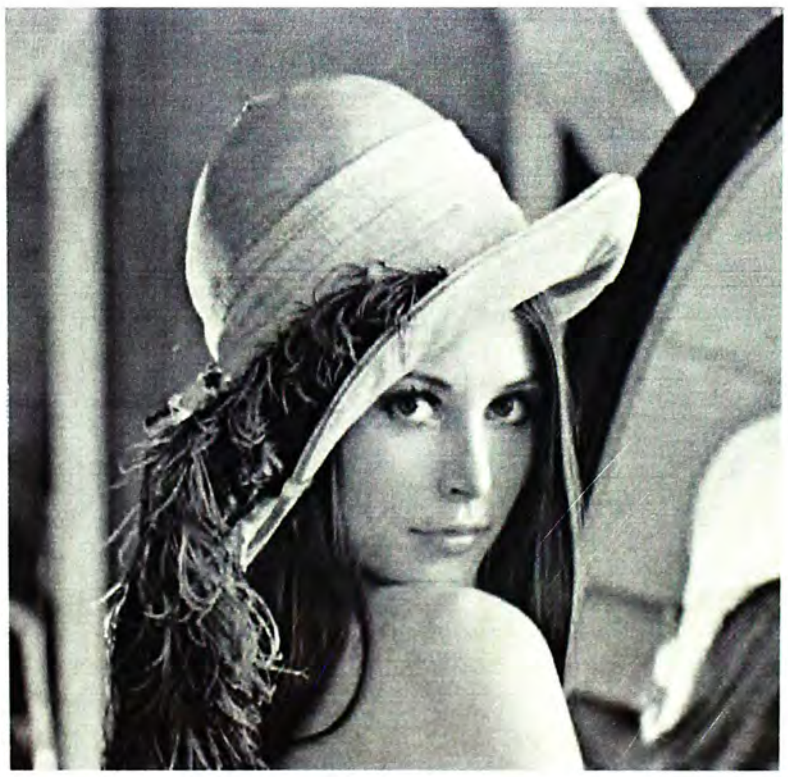


Figure 3.7: Decoded Lena image at total bit rate 1.0 at receiver side after passing through channel

From Figure 3.8 to Figure 3.11, they show the decoded Lena images at various total bit rate 0.2, 0.3, 0.5, and 1.0 using non-adaptive method and adaptive method respectively at receiver side after passing through channel with BER = 0.1. Comparing these figures, we can find the quality of output images using adaptive method is better than output images using non-adaptive method at the same bit rate.

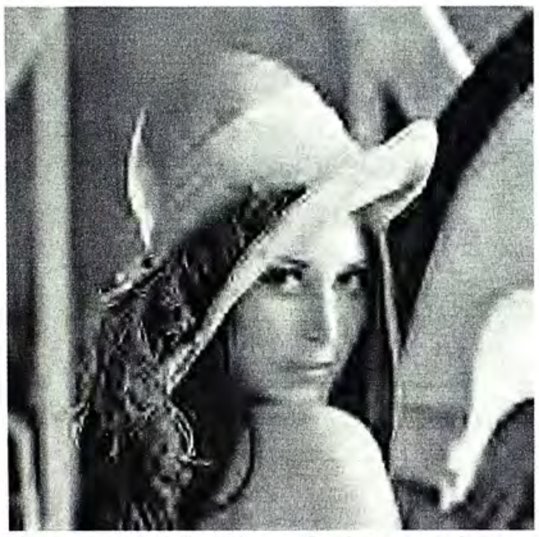


Figure 3.8(a): Decoded Lena image at total bit rate 0.2 with non-adaptive method

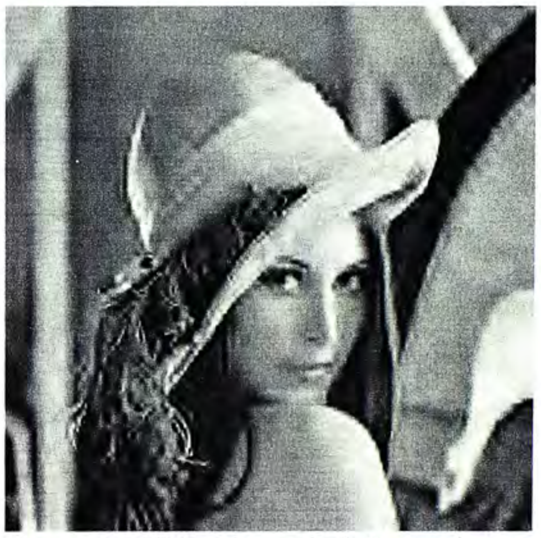


Figure 3.8(b): Decoded Lena image at total bit rate 0.2 with adaptive method

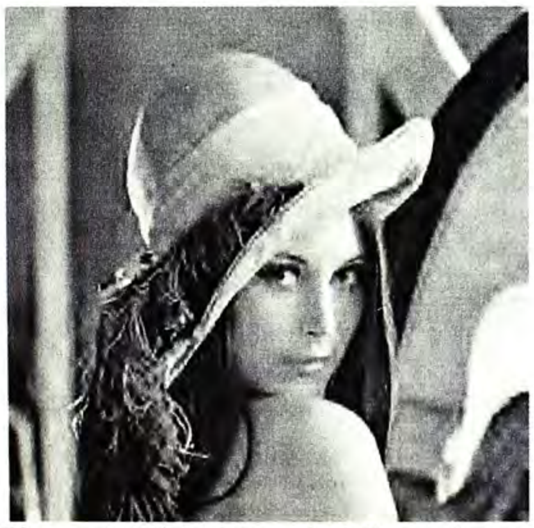


Figure 3.9(a): Decoded Lena image at total bit rate 0.3 with non-adaptive method

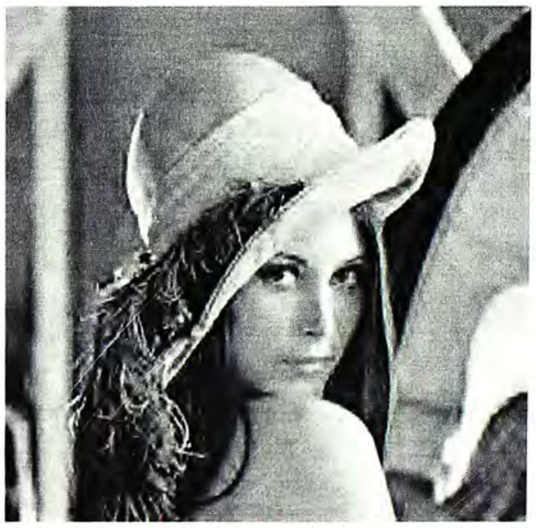


Figure 3.9(b): Decoded Lena image at total bit rate 0.3 with adaptive method

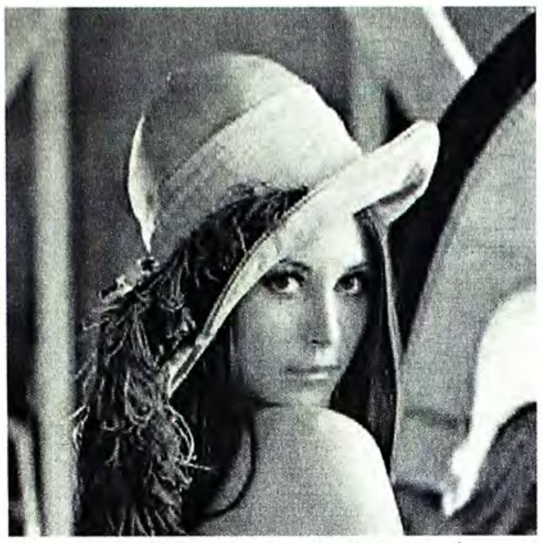


Figure 3.10(a): Decoded Lena image at total bit rate 0.5 with non-adaptive method

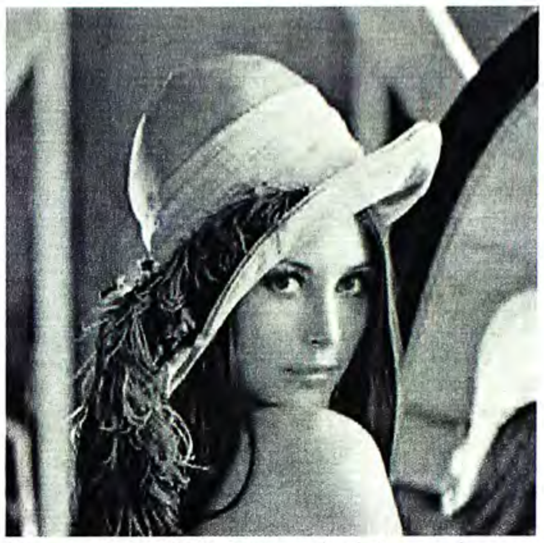


Figure 3.10(b): Decoded Lena image at total bit rate 0.5 with adaptive method

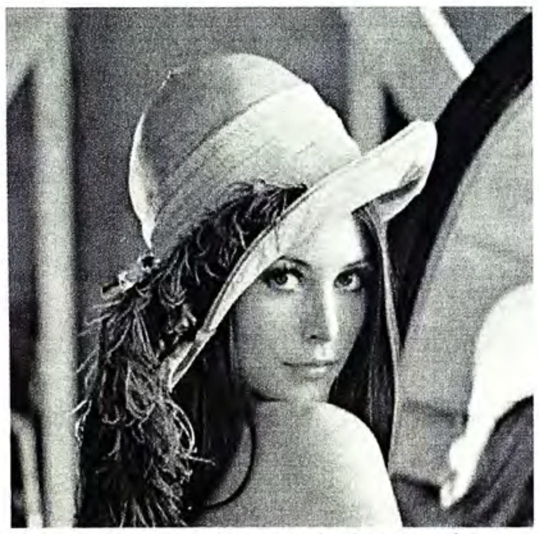


Figure 3.11(a): Decoded Lena image at total bit rate 1.0 with non-adaptive method

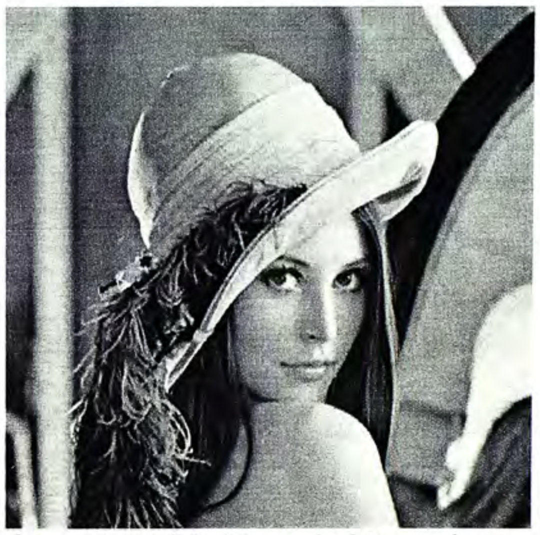


Figure 3.11(b): Decoded Lena image at total bit rate 1.0 with adaptive method

Chapter 4

Unequal-Error-Protection Provided by Trellis-Coded Modulation

An unequal error-protection (UEP) coding scheme using trellis-coded modulation (TCM) is proposed for a single data stream consisting of information with two levels of importance. UEP is provided by switching between two constellations, fine and coarse constellation. The coarse constellation can be derived from the fine constellation easily. This coding scheme can be decoded by the Viterbi algorithm by traversing the trellis with unequal constellation. The simulation finds that the error rate of the important information is lower than the error rate for an equivalent equal error-protection code at the same data rate but the average performance in term of probability of error is only slightly worse than the performance of equal protection scheme. Compared with the conventional equal error protection, unequal error protection is provided to a data source without changing the data rate and increase of system complexity.

4.1 System Description

The proposed UEP coding system shown in the Figure 4.1 is a modification version of the time-multiplexing approach mentioned in [17] and [18]. The input data is assumed to consist of two importance levels. In Fig. 4.1, *H* is used to denoted important data and the *L* is used to denote less important data. The important data and less important data are multiplexed to form a single data stream. Stream pattern is looked like *HLHLHL*... or *LHLHLH*.... The data stream is fed into the TCM encoder using the unequal constellation to provide unequal error protection. The constellation with larger free distance is used to transmit important data and the constellation with smaller free distance is used to transmit less important data. At the receiver side, Viterbi algorithm is used to decode received data and then de-multiplex the decoded data into important and less important data.

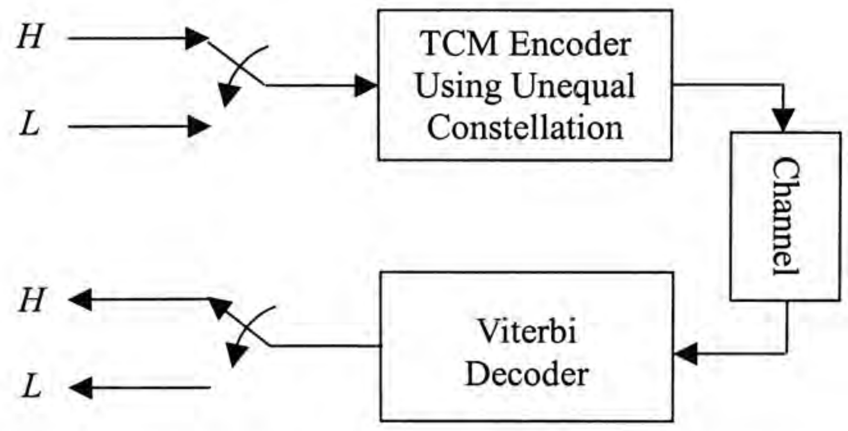


Figure 4.1: The proposed UEP system using unequal constellation for TCM

4.2 Unequal Constellation

Two signal constellations, fine and coarse constellation, are adopted to achieve unequal error protection for TCM. F is used to denote the fine constellation and C is used to denote the coarse constellation. Coarse constellation can be derived from the fine constellation easily and obtained by mapping every two adjacent constellation points in the fine constellation to one point in the coarse constellation.

Assume that there are 4 constellation points for the fine constellation {0, 1, 2, 3}. The coarse constellation has two constellation points {0, 1} by mapping {0, 1} and {2, 3} from fine constellation to {0} and {1} respectively, as illustrated in Figure 4.2. It is obvious that the free distance of coarse constellation is always larger than fine constellation for the same transmission power. Coarse constellation is used to encode the important data while fine constellation is used to encode less important data in order to provide more protection to important data.

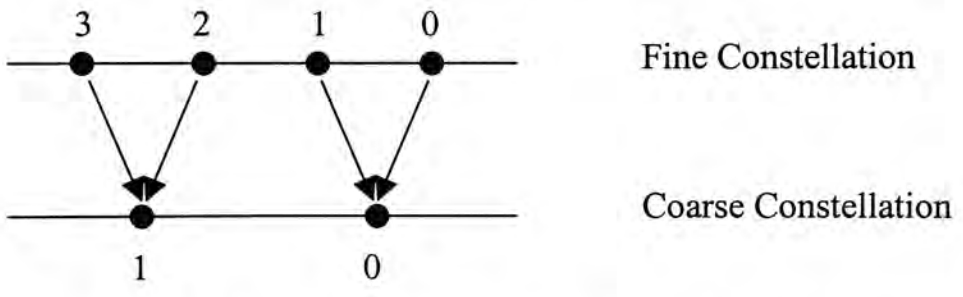


Figure 4.2: Fine constellation and coarse constellation

The encoding processing with unequal constellation is identical to the ordinary TCM scheme since both fine and coarse constellations have the same trellis structure.

Viterbi algorithm can be used to trace the trellis diagram to find the sequence with minimum metric. The discrepancy with ordinary TCM scheme is the usage of two constellations instead of one constellation for the whole encoding and decoding process in the ordinary TCM scheme.

Without loss of generality, in later part of this chapter, we consider the special case that fine and coarse constellations are used alternatively time after time with equal probability. Figure 4.3 shows one fine constellation followed by another coarse constellation and vice versa. The pattern is looked like *FCFCFC....* Without loss of generality, fine constellation is used at even stages and coarse constellation at odd stages. In general, unequal constellation scheme can be applied to any permutation

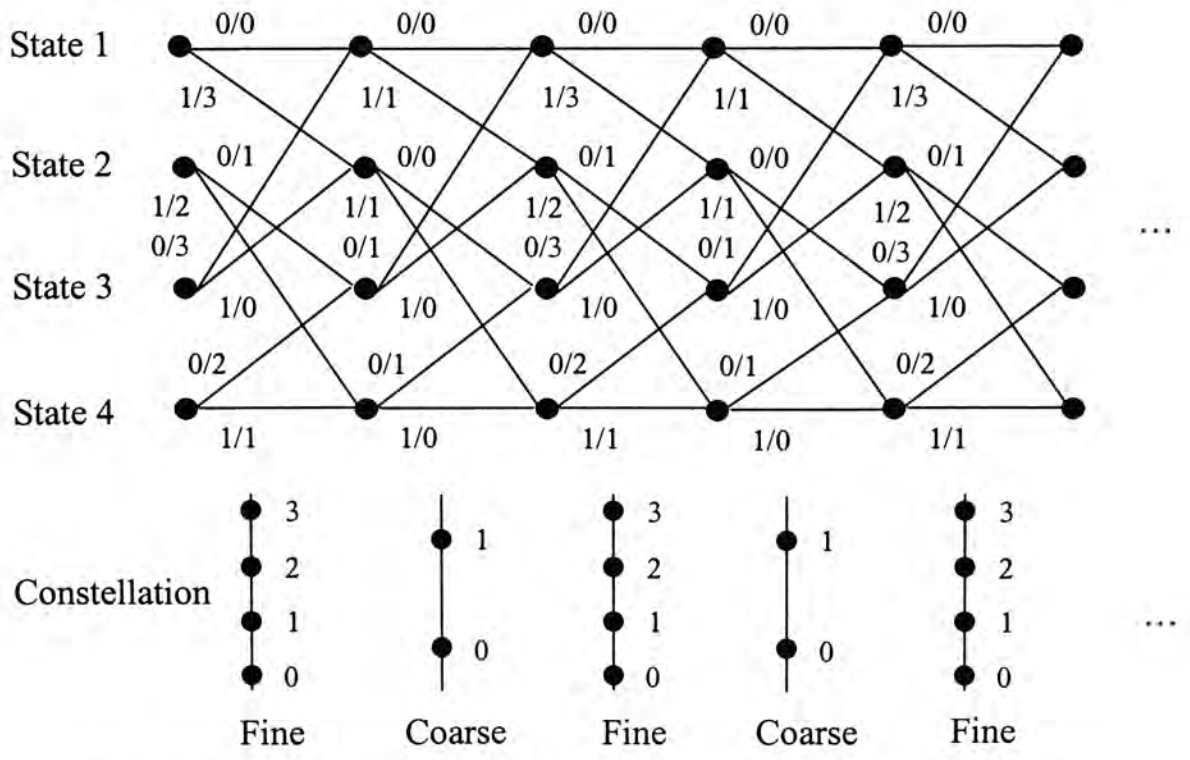


Figure 4.3: Trellis diagram of (2, 1, 2) TCM using unequal constellation

and ratio of fine and coarse constellation. If the proposed scheme used the fine constellation only, it becomes an ordinary TCM scheme without unequal error protection property.

4.3 Free Distance

To obtain the free distance d_{free} of the ordinary TCM scheme, the algorithm derived by Saxena [19] and Mulligan and Wilson [20] is used, following the method of [10]. The free distance of the ordinary TCM scheme is the minimum Euclidean distance between a pair of path forming an error event during traversing the trellis [10]. For the trellis describing the TCM using unequal constellation scheme, every pair of branches in a section of the trellis defines one distance between the signals labeling the branches. The squared Euclidean distance between the signal sequences associated with two paths in the trellis is obtained by summing the individual squared Euclidean distances. If parallel transitions occur, every branch is associated with an entire subconstellation, only the minimum distance between any two signals extracted from the pair of subconstellations will be used.

The algorithm is based on the update of the entries of a matrix $D^{(n)} = (\delta_{pq}^{(n)})$, which are the minimum squared Euclidean distances between all pairs of paths diverging from the same initial state and reaching the states p and q at discrete time n. Two such pairs of paths are shown in Figure 4.4. The matrix $D^{(n)}$ is symmetric with dimension S×S, and that its elements on the main diagonal are the distances between

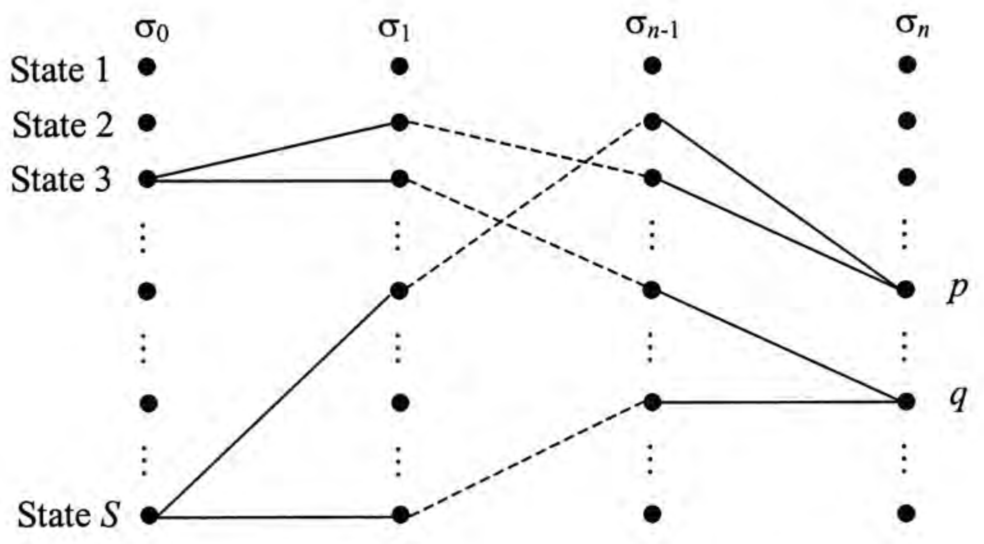


Figure 4.4: Two pairs of paths diverging at time $n = 0$ and reaching the states p, q at the same time

remerged paths ("error events"). When all entries δ_{pq} in the matrix are larger than or equal to $\min_{p}(\delta_{pp}^{(n)})$, the algorithm will be terminated and the free distance is declared to be equal to $\min_{p}(\delta_{pp}^{(n)})$.

Because both fine and coarse constellations in our proposed UEP scheme have the same trellis structure as ordinary TCM, the above algorithm can also be used to find the free distance for TCM using unequal constellation. The free distance is presented in the Table 4.1 and Table 4.2 for 4-AM and 8-AM respectively, whose parity check matrix is adopted from [21]. The difference of parity check matrix between 4-AM and 8-AM is that 8-AM has one more zero row, which results in the parallel transition. From tables, the free distance of 4-AM with both equal constellation and unequal constellation is same to 8-AM. However, free distances of TCM using unequal constellation for both equal constellation and unequal constellation with pattern *FCFC*... and *CFCF*..., are smaller than the free distances of TCM using equal constellation. The block diagram of the encoder of 4-AM, 4 states TCM is shown in Figure 2.11.

No of State	Parity check matrix	d^2_{free} for equal constellation	d^2_{free} for unequal constellation (with pattern *FCFC*...)	d^2_{free} for unequal constellation (with pattern *CFCF*...)
4	$\begin{bmatrix}1&0&1\\0&1&0\end{bmatrix}$	9	6	7
8	$\begin{bmatrix}1&0&1&1\\0&1&0&0\end{bmatrix}$	10	7	7
16	$\begin{bmatrix}1&0&0&1&1\\0&0&1&0&0\end{bmatrix}$	11	7	8
32	$\begin{bmatrix}1&0&0&1&0&1\\0&0&1&0&0&0\end{bmatrix}$	13	7	8
64	$\begin{bmatrix}1&0&0&0&0&1&1\\0&0&1&0&1&0&0\end{bmatrix}$	14	8	8
128	$\begin{bmatrix}1&0&0&1&1&1&0&1\\0&1&0&1&0&1&1&0\end{bmatrix}$	16	10	9
256	$\begin{bmatrix}1&0&1&0&0&1&1&0&1\\0&1&1&1&1&0&0&1&0\end{bmatrix}$	17	10	9

Table 4.1: Free distance for equal constellation and unequal constellation using 4-AM

No of State	Parity check matrix	d_{free}^2 for equal constellation	d_{free}^2 for unequal constellation (with pattern *FCFC*...)	d_{free}^2 for unequal constellation (with pattern *CFCF*...)
4	$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	9	6	7
8	$\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	10	7	7
16	$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	11	7	8
32	$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	13	7	8
64	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	14	8	8
128	$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	16	10	9
256	$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$	17	10	9

Table 4.2: Free distance for equal constellation and unequal constellation using 8-AM

4.4 Simulation Results

In the simulation, two uniformly distributed sequences are generated to model the important data and less important data and they are passed to the TCM encoder with unequal constellation. The pattern of unequal constellation is *FCFCFC*.... The modulated signals are transmitted through the AWGN channel. The receiver receives corrupted signal from channel and uses the Viterbi algorithm to find the best sequence as decoded output by traversing the trellis with unequal constellation. Thousand of simulations were conducted to obtain the probability of error versus channel signal-to-noise ratio (SNR).

Figure 4.5, 4.6 and 4.7 show probability of error versus SNR for 4-AM TCM scheme with 8 states, 4-AM TCM scheme with 16 states and 8-AM TCM scheme with 8 states respectively. The performance in term of error probability of coarse constellation is better than the equal constellation. It has 0.3-0.5 dB SNR improvements for same error probability like 10^{-5}. However, the performance for fine constellation is worse than equal constellation. With unequal constellation, the performance of fine constellation is sacrifice to enhance the error correcting ability of coarse constellation. Thus, the unequal constellation can provide two level of unequal error protection. The coarse constellation can be used to encode the important data while the fine constellation can be used to encoder the less important data.

Besides, the performance of average error probability of the TCM using unequal constellation is slightly worse than TCM using equal constellation. In Table 4.1 and 4.2, the free distance of unequal constellation is smaller than the free distance of equal constellation and the probability of error is related to free distance. The larger free distance a code has, the smaller probability of error it will obtain. In the figures, they also show the probability of error for uncoded 2-AM and 4-AM which have same data rate as 4-AM TCM and 8-AM TCM respectively. Let P_c, P_e, P_o, P_f and P_u be the probability of error for coarse constellation, equal constellation, both coarse and fine constellation, fine constellation, and uncoded AM scheme with same data rate respectively. Then the relation between them is:

$$P_c < P_e < P_o < P_f < P_u . \tag{4.1}$$

Therefore, this coding system can provide unequally error-protected information with two levels of importance without changing the data rate and increase of system complexity. The coarse constellation is used encode the important data while fine constellation is used to encode less important data. As a result, it can provide more protection for the important data and probability of error for the important data is lower than equal protection scheme.

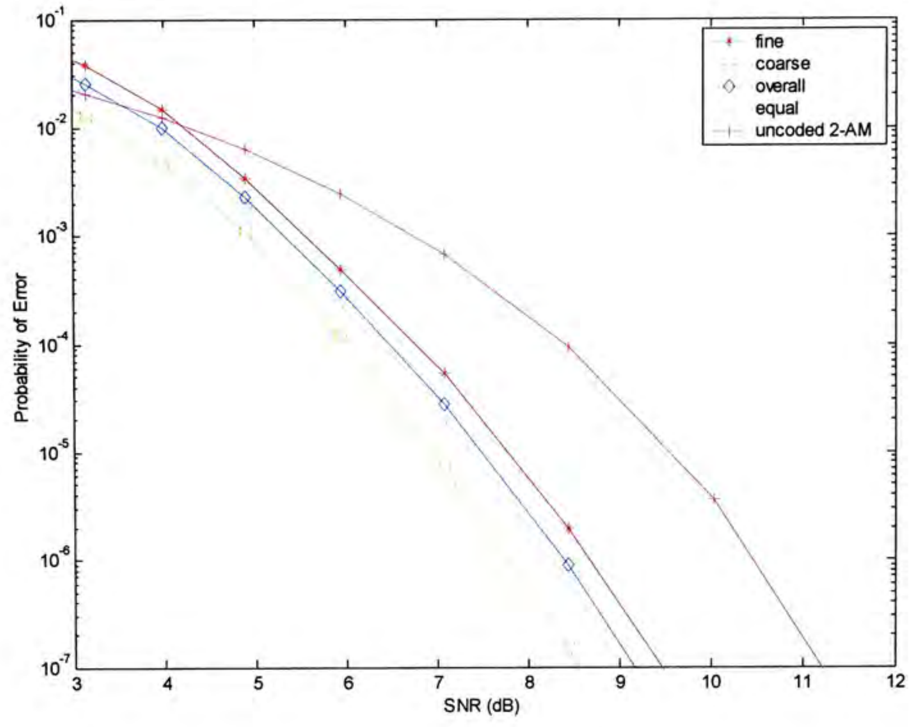


Figure 4.5: Probability of error to SNR for 4-AM, 8 states TCM using unequal constellation

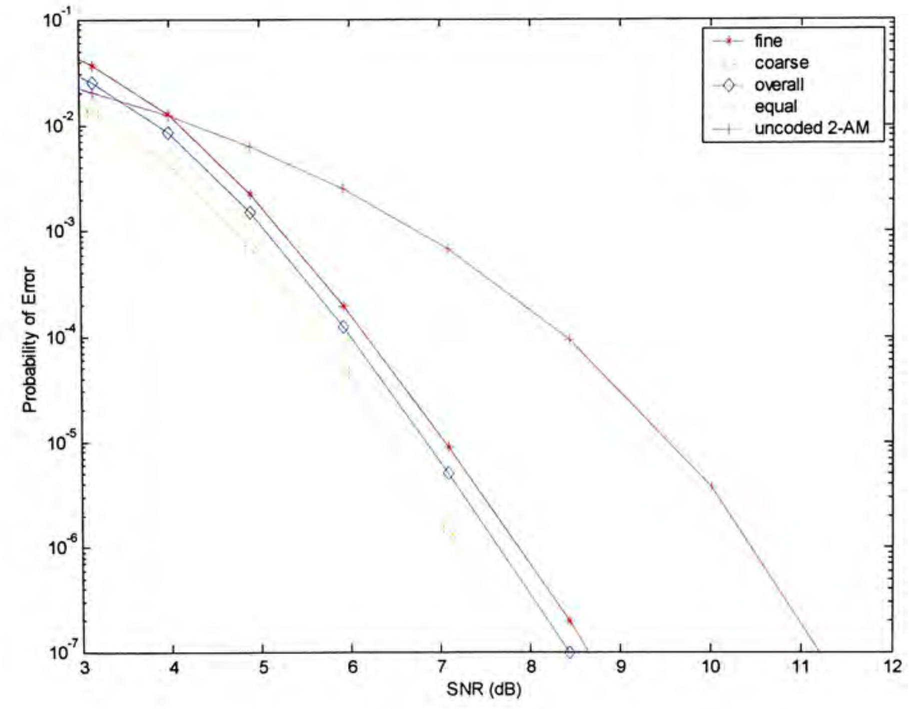


Figure 4.6: Probability of error to SNR for 4-AM, 16 states TCM using unequal constellation

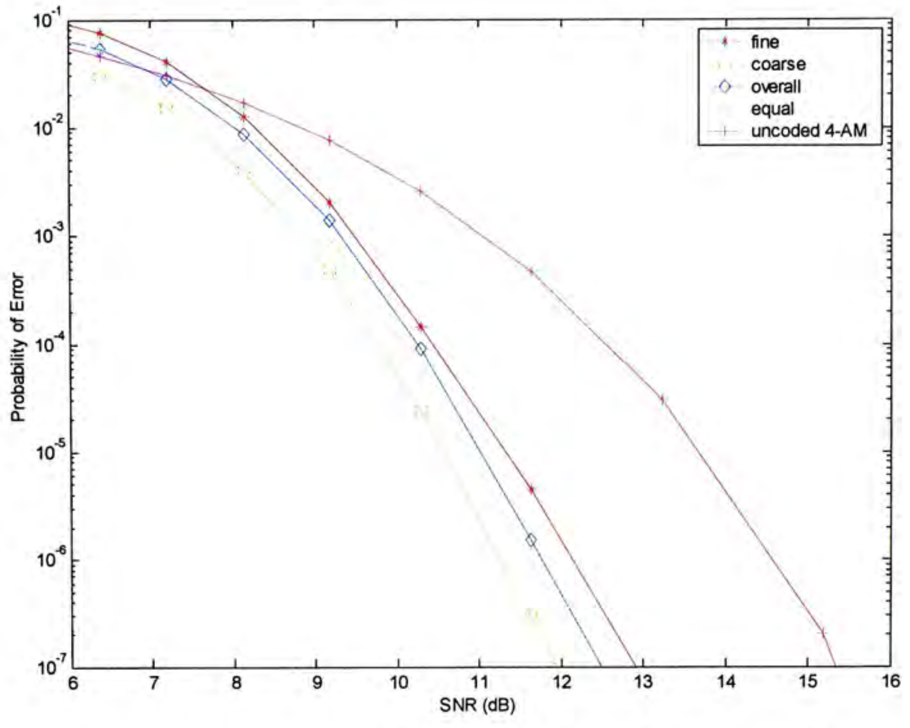


Figure 4.7: Probability of error to SNR for 8-AM, 8 states TCM using unequal constellation

Chapter 5

Conclusion

This thesis derives a simple and low complexity algorithm for channel codes allocation to provide unequal error protection for SPIHT coded image transmitted over noisy binary symmetric channels. It has a complexity of only $O(N)$, which is far smaller than other channel allocation algorithms for SPIHT coder. The performance of the algorithm is simulated on standard 512×512 gray scale Lena image with BER=0.1. Numerical result shows our proposed algorithm can yield 0.4 to 0.8 dB performance improvements in PSNR over the equal error protection scheme.

This thesis also invented an unequal error protection code based on trellis-coded modulation (TCM). It can unequally protect information with two levels of importance. The free distances of new UEP code using AM modulation is obtained by the simulation. Furthermore, probability of error versus SNR for new UEP code over AWGN is also simulated. The numerical result shows that the proposed UEP code's overall performance in term of probability of error is only slightly worse than the performance of equal protection scheme for the same code rate, but it can provide more protection for the important data and probability of error for the important data is lower than equal protection scheme.

Bibliography

[1] P. Elias, "Coding for Noisy Channels", *IRE Conv. Rec.*, Part 4, pp.37-47, 1955.

[2] J. M. Wozencraft and B. Reiffen, *Sequential Decoding*, MIT Press, Cambridge, Mass., 1961

[3] J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Trans. Inf. Theory*, IT-13, pp. 260-269, April 1967.

[4] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, 1983

[5] M. Bossert, *Channel Coding for Telecommunications*, Wiley, 1999.

[6] J. B. Cain, G.C. Clark, and J.M. Geist, "Punctured convolutional codes of rate $(n\text{-}1)/n$ and simplified maximum likelihood decoding," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 97-100, Jan. 1979.

[7] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," *IEEE Trans. Commun.*, vol. 36, no. 4, pp. 389-400, 1988.

[8] G. Ungerböck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inform. Theory*, Vol. IT-28, No. 1, pp. 55-67, 1982

[9] J. G. Proakis, M. Salehi, *Communication Systems Engineering*, Prentice Hall, 1994

[10] E. Biglieri, D. Divsalar, P. J. McLane, M. K. Simon, *Introduction to Trellis-Coded Modulation with Application*, Macmillan Publishing Company,

1991.

[11] N. Seshadri and C.-E. W. Sundberg, "List Viterbi decoding algorithm with applications," *IEEE Trans. Commun.*, vol. 42, no. 2-4, pp. 313-323, 1994.

[12] A. Said and W. A. Pearlman, "A new, fast, and efficient image codes based on set partitioning in hierarchical trees," *IEEE Tans. Circuits and System for Video Technology*, vol. 6, no. 3, pp. 243-250, 1996.

[13] P. G. Sherwood and K. Zeger, "Progressive image coding for noisy channels," *IEEE Signal Processing Letters,* vol. 4, no. 7, pp. 189-191, 1997.

[14] V. Chande and N. Farvardin, "Joint source-channel coding for progressive transmission of embedded source coders," *Proc. of Data Compression Conference,* pp. 52 –61, 1999.

[15] M. S. Ho, C. F. Kwong, K. H. Chei and K. P. Ho, "Channel code allocation for unequal error protection to embedded image coders ", *Asia Pacific Conference in Commun.*, pp. 1098-1011, Nov. 2000.

[16] J. M. Shappiro, "Embedded image coding using zerotrees of wavelet coefficients," IEEE Transactions on Signal Processing, vol. 41, no. 12, pp. 3445-3462, Dec, 1993.

[17] A. R. Calderbank and N. Seshadri, "Multilevel codes for unequal error protection", *IEEE Trans. Infom. Theory*, vol. 39, pp. 1234-1248, July 1993

[18] L. F. Wei, "Coded modulation with unequal error protection," *IEEE Trans. Commun.*, vol. 41, pp. 1439-1449, Oct. 1993

[19] R. C. P. Saxena, "Optimum encoding in finite state coded modulation," Report TR83-2, Department of Electrical, Computer and System Engineering, Rensselaer Polytechnic Institute, Troy, N.Y., 1983.

[20] M. M. Mulligan and S. G. Wilson, "An improved algorithm for evaluating trellis phase codes," *IEEE Trans. Inf. Theory*, vol. IT-30, No. 6, pp.846-851, Nov. 1984.

[21] G. Ungerboeck, "Trellis-coded modulation with redundant signal sets. Part II: State of the art," *IEEE Commun. Mag.*, Vol. 25, pp. 12-22, Feb.1987.