JOINT SOURCE-CHANNEL DECODING OF VARIABLE-LENGTH ENCODED SOURCES WITH APPLICATIONS TO IMAGE TRANSMISSION

by

K.P. Subbalakshmi

B.Sc., Physics, University of Madras, 1990

M.E., E.C.E, Indian Institute of Science, 1994

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Engineering Science

SIMON FRASER UNIVERSITY
July 2000

APPROVAL

Name: K.P. Subbalakshmi

Degree: Doctor of Philosophy

Title of thesis: Joint Source-Channel Decoding of Variable-Length Encoded Sources with Applications to Image Transmission

Examining Committee: Professor Mehrdad Saif, Chairman

Associate Professor Jacques Vaisey
Senior Supervisor, School of Engineering Science

Professor James Cavers
School of Engineering Science

Professor Paul Ho
School of Engineering Science

Professor Ze-Nian Li
School of Computing Science

Assistant Professor Amir Asif
Technical University of British Columbia

Date Approved: _____

Abstract

Shannon's source-channel separation theorem holds only under asymptotic conditions, where both source and channel codes are allowed infinite length and complexity, which is not possible in practice. This observation has led to the increasing popularity of joint source-channel encoding and decoding schemes as viable alternatives for achieving reliable communication of signals across noisy channels. Joint source-channel encoders (JSCE) aim at designing the source and channel *encoder* of a system in some joint sense, while joint source-channel *decoders* (JSCD) concentrate on the design of the decoder.

JSCD schemes have been well studied for sources which are *not* variable-length encoded. However, it is well known that entropy codes offer significant improvement in the noiseless rate-distortion performance of a system, and that most entropy codes are variable length in nature. Hence, designing JSCDs for variable-length encoded sources is of interest. This dissertation presents one of the first solutions to the problem of designing optimal, maximum a posteriori probability (MAP) decoders for different variable-length encoded sources over channels without memory and the first such solution for channels *with* memory. Also, the complexity of the solutions presented in this dissertation remains a constant with time unlike that of the other solutions. Moreover, this dissertation presents the first application of the MAP-BSC algorithm to a subband based image coding system.

In the case of fixed length codes (with N codewords), the MAP problem reduces to processing $\log_2(N)$ bits at one time. This is not a good solution when the codewords are of unequal lengths. When errors are introduced by the channel in a variable-length encoded signal, the partition of the bit stream into component codewords is no longer obvious. Hence one must find the "best" possible parsing of the bit stream using some measure of "goodness". Finding such an optimal solution involves a search over

all possible partitions of the bit stream which is computationally expensive; so we provide a dynamic programming solution to the problem so as to limit the complexity of the search. One of the main contributions of this dissertation is the introduction of the notion of *complete* and *incomplete* states in the state space associated with the dynamic programming formulation which lets us deal with the variable-length codewords in an elegant manner.

In the first part of this dissertation, we consider the design of the optimal MAP decoder for variable-length encoded sources over binary symmetric channels. We first consider Markov sources and then particularize it to memoryless sources. We also show how the proposed algorithm may be extended to Markov sources of higher orders. The second part of the dissertation proposes the design of the MAP decoder for variable-length encoded sources over channels with memory. Performance of both the decoders are compared to that of the standard decoders to demonstrate the superiority of the proposed decoders over the conventional ones. The robustness of the decoders under channel mismatch conditions is demonstrated experimentally. In the third part of the dissertation, the proposed decoder is tested on an image transmission system to show that the decoder performs significantly better than the conventional decoder both perceptually and objectively (in terms of the peak signal-to-noise ratio) even when the source is not "cleanly" modeled. Other salient features of the algorithms developed in this dissertation are that no assumptions are made on the number of samples or bits transmitted.

Acknowledgments

At the outset, I would like to thank my senior supervisor, Prof. Jacques Vaisey, for his support and guidance throughout this project. I would like to thank Prof. James Cavers and Prof. Paul Ho for being on my supervisory committee, Prof. Ze-Nian Li for being my internal examiner and Prof. Amir Asif for being my external examiner.

Thanks are also due to the efficient members of the system support team at SFU; particularly, Stephen Chan, Chao Cheng, Frank Manuel and Alfred Ng for their friendly and timely help with all matters pertaining to computers and to Marilyn Anderson, Jackie Briggs, Brigitte Rabold and Annie Radisic for keeping my life at SFU free of administerial hiccups.

Thanks to Yu Chen, Ed Chiu, Chen Ji, Yingbo Jiang, Tong Jin, Ming Li and Jerry Zhang for making the lab lively and for the Chinese lessons. Thanks to Anoja and Pri for making my stay at Burnaby memorable. Thanks to Kalpagam and family for all the good times and the great food. Thanks to my husband, Mouli, and to Uma, for a decade of friendship. Thanks to my sisters, Sujatha and Sunanda, for a lifetime of friendship. Thanks to my parents, who taught me to learn. Thanks to British Columbia, for being beautiful!

Thanks are also due to the ECpE department of Iowa State University for letting me use their computers towards the final stages of writing this dissertation.

Dedication

To

Paatti, Appa, Amma, Sujatha and *Sunanda*

Contents

List of Tables

List of Figures

List of Abbreviations

AMC	:	Additive Markov Channel
AR(1)	:	First order auto-regressive
BFOS	:	Breiman Friedman Olshen Stone
bpp	:	Bits per Pixel
BSC	:	Binary Symmetric Channel
DWT	:	Discrete Cosine Transform
DWT	:	Discrete Wavelet Transform
ECVQ	:	Entropy Constrained Vector Quantizer
EOL	:	End of Line
FIR	:	Finite Impulse Response
FLC	:	Fixed Length Codes
gcd	:	Greatest Common Divisor
HD	:	Huffman Decoder
HP-band	:	High-pass bands in a wavelet decomposition system
JPEG	:	Joint Photographic Experts Group
JSCC	:	Joint Source Channel Coding
JSCD	:	Joint Source Channel Decoder
JSCE	:	Joint Source Channel Encoder
KLT	:	Karhunen-Loève Transform
LL-band	:	Low-Low band in a wavelet decomposition system
MAP	:	Maximum a posteriori Probability
MAP-BSC	:	Maximum a posteriori Probability decoder for transmission over BSCs
MAP-AMC	:	Maximum a posteriori Probability decoder for transmission over AMCs
MSB	:	Most Significant Bit

MSNR	:	Modified Signal to Noise Ratio
PBOS	:	Percentage of Bits Out of Synchronization
pdf	:	Probability Density Function
PR	:	Perfect Reconstruction
PSNR	:	Peak Signal to Noise Ratio
SPIHT	:	Set Partitioning in Heirarchical Trees
SR	:	Stochastic Relaxation
VLC	:	Variable Length Codes
VQ	:	Vector Quantizer

Chapter 1

Introduction

A fundamental problem in communication theory is that of achieving reliable transmission of data from one point to another, given some measure of reliability between the source at the transmitter end and its reproduction at the receiver end. Shannon (1948) showed that a discrete memoryless channel has a limit on the maximum amount of information, called the capacity of the channel, that can be transmitted over this channel without incurring any distortion. Subsequently (Shannon 1959), he also showed that given a discrete, stationary and ergodic source and a single letter distortion measure there is a rate-distortion function that can be interpreted as the minimum information rate of the source for a given level of distortion. These two results form the backbone of two main branches of communication research: *source coding* and *channel coding*. In source coding, the aim is often to *remove* the redundancy in the source so that the source can be represented with fewer bits; in essence,the aim is to approach the *rate-distortion* (R-D) bound as closely as possible. In channel coding, the dual of the source coding problem, the aim is to protect the source from the errors that are introduced by the channel. This goal is often achieved by the *introduction* of controlled redundancy. These two conflicting requirements often lead to an engineering compromise when selecting source and channel codes for some rate criterion. In this dissertation we will discuss a problem which lies in an area that straddles the two types of coding.

1.1 Motivation

Even though Shannon (1948) showed that source and channel codecs can be optimized individually and then operated in a cascaded system without any sacrifice in optimality, it has been shown that this is not true for all channels (Vembu, Verdu, and Steinberg 1995). Also, where this is valid, it is valid only under asymptotic conditions, where the source and channel codes are allowed arbitrarily large lengths. In terms of implementation, this means arbitrary complexity and delay. Hence, in most practical systems, where infinite delay and complexity are not possible, designing the two codecs separately might not allow us to approach the R-D bound closely.

The above observation has led to techniques that design the source and channel codecs simultaneously. One may classify these joint design techniques into three very broad categories: *joint source-channel encoders* (JSCE), *joint source-channel decoders* (JSCD) and *rate allocation strategies*. As the name implies, JSCEs are essentially source or channel *codes* that have been designed using some knowledge of the channel and source characteristics and the JSCDs are *decoders* that are designed in some joint sense and where no real modification is made to the codes themselves. In the last category, the source and channel codes are designed separately and operated in tandem, but the bits allocation to each coder is done jointly.

Further, it is well known that the use of entropy codes give better performance in the R-D sense in a *noiseless* environment. In fact, most practical image and video coding standards, like JPEG (Wallace 1992) and MPEG (LeGall 1991), recommend Huffman (1952) (or arithmetic) codes. Since most practical channels are noisy, it is worthwhile to explore joint source-channel decoders for entropy codes. However, most entropy codes are variable length codes. Hence, the problem of designing JSCDs for entropy codes boils down to designing JSCDs for variable length codes. Dealing with variable length codes in a noisy environment becomes a challenging problem, since a single bit error introduced by the channel can lead to a series of word decoding errors, which throw the transmitter and the receiver out of synchronization. In Chapter 2 we will discuss the issues involved in dealing with variable length codes in detail.

Good source codes try to remove the redundancy present in the source; however, in a practical system, there is often significant residual redundancy left in the source even after source coding. The aim of this dissertation is to design JSCDs that will

exploit this residual redundancy in the source, to achieve an overall performance improvement over the decoders that do not make use of it. The residual redundancy in the source can be considered to be of two types: that due to memory and that due to the non-uniformity in the pdf of the source (Phamdo 1993). The first kind of redundancy arises when the memory in the source is not completely removed by the source coder. The second type of residual redundancy arises when the pdf of the source is not flat. When the pdf of the source is not flat, some of the symbols occur more frequently than others and knowing this can help the decoder recover from some of the errors introduced by the channel.

In this dissertation we focus on developing joint source-channel decoding algorithms for variable-length encoded sources with and without memory to be transmitted over both *binary symmetric channels* (BSC) and *additive Markov channels* (AMC). In particular, we aim at developing a decoder that minimizes the probability of error between the transmitted and received sequence, given the knowledge of the source and channel statistics. In other words, we design a maximum *a posteriori* probability (MAP) decoder for entropy coded sources transmitted over noisy channels. We also show that the decoder designed for the BSCs can be incorporated in an image transmission system to increase the performance of the codec substantially.

1.2 Scope of the Work

In order to prevent the catastrophic effects of single bit errors, entropy coded information is usually protected using channel codes. It is therefore of interest to look into the performance of the proposed decoder for channel encoded streams. It is potentially possible to combine the channel decoder and the MAP decoder into one JSCD. This is especially facilitated by the proposed state-space structure, since the state-space of a trellis based convolutional decoder can, conceivably, be combined with the proposed one. One can further enhance the performance of such an overall system by optimally allocating the bits between the source and channel codes. In this dissertation, however, we do not consider bit allocation issues.

Error resilience can be achieved in a myriad of ways, some of which will be discussed in the next chapter. In this dissertation the main aim is to optimize the decoder in a probabilistic sense. Designing channel matched source *encoders* that can then be

used in tandem with the proposed scheme, is left for future work. Since optimizing the encoders and the decoders are orthogonal procedures, one can hope for an overall increase in system performance when these techniques are used in conjunction with each other.

1.3 Original Contributions of the Dissertation

As stated earlier, the main problem arising in the treatment of entropy coded sources is the variable length nature of the codes. In the case of fixed length codes with N codewords, the MAP problem reduces to processing $\log_2(N)$ bits at one time. This is not a good solution when the codewords are of unequal lengths. When errors are introduced by the channel in a variable-length encoded stream, the partition of the bit stream into component codewords is no longer obvious, since it is possible for a codeword to split into more than one codeword or for two or more codewords to merge together to form a different codeword (see Section 2.3). Hence one must find the "best" possible parsing of the bit streams using some measure of "goodness". Finding the optimal solution involves a search over all possible partitions of the bit stream, which is time consuming. So, we solve the problem using dynamic programming to reduce the complexity of the search and to guarantee a finite expected delay in decoding the transmitted sequence. One of the main contributions of this dissertation is the introduction of the notion of *complete* and *incomplete* states (cf. Chapter 3) in the state-space associated with the dynamic programming formulation. The proposed state-space lets us deal with the codewords in the codebook in an elegant manner and makes the expected delay of the decoding operation finite.

In the first part of this dissertation, we will consider the design of the optimal MAP decoder for entropy coded sources over BSCs. We first solve the problem for Markov sources (Subbalakshmi and Vaisey 1999a) and then particularize it to memoryless sources (Subbalakshmi and Vaisey 1998). This is followed by a demonstration of the optimality of these algorithms. We also show how the proposed algorithm may be extended to Markov sources of higher orders. The second part of the dissertation involves the design of the MAP decoder for entropy coded sources over channels with memory (Subbalakshmi and Vaisey 1999b). Performance of both the decoders are compared to that of standard decoders and it is shown that the proposed decoder does

significantly better than the standard ones. The robustness of the proposed decoders under channel mismatch conditions is demonstrated experimentally. In the third part of the dissertation, we test the proposed decoder in a more practical environment. Specifically, we apply it to image transmission along with an encoder scheme that involves the use of end-of-line (EOL) symbols to show that the decoder performs significantly better than the conventional decoder both perceptually and objectively (in terms of the peak signal-to-noise ratio).

Other salient features of the algorithms developed in this dissertation are that their complexity remains a constant with time and that there is no assumption made of the number of samples or bits transmitted. The state-space proposed for the MAP decoder for Markov sources over BSC can be directly used for an AMC as well.

1.4 Organization of the Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, the mathematical concepts behind MAP decoders are briefly introduced. The issues involved in transmitting variable-length encoded signals over noisy channels are discussed and an overview of the existing methods to tackle the problem is presented. We also describe some common building blocks in source coding systems, which will be used to build coders for testing the proposed decoders. In Chapter 3 we develop the MAP decoder for variable-length encoded Markov sources and memoryless sources transmitted over BSC. We also look at a general formulation of the problem for Markov sources of higher orders. This decoder is then extended to Markov sources transmitted over channels with memory in Chapter 4. Applications to image transmission are discussed in Chapter 5.

Chapter 2

Background and Related Work

In this chapter we will present a brief overview of the mathematical tools that will be used in the dissertation. A literature review of the existing error-resilient coding techniques, with special emphasis on joint source-channel codec design algorithms is then presented. This is followed by a look at the issues arising in the transmission of variable length codes over noisy channels and some performance measures that will be used to evaluate the algorithms proposed in this dissertation. Finally, some of the components in a generic signal compression scheme will be presented, which will be used in the forthcoming chapters to test the proposed decoders.

2.1 Maximum a posteriori Probability Decision Rule

Statistical decision theory (Poor 1988) is a basic tool that is used in many areas of communications, control theory and system theory. The essential structure of a problem in decision theory contains four stages: an event occurs, this is relayed to an observer through some signal, a noisy observation of this message is made by the observer and finally a decision is taken regarding the message that was sent. Generally, the set of possible messages is called the *message space*, M; the set of possible signals is called the *signal space*, S; the set of possible observations is called the *observation space*, Z; and the set of all decisions possible is called the *decision space*, D.

When the set of points in the message space is finite, the problem is said to be a decision theoretic one, and when it is uncountably infinite, then the problem becomes

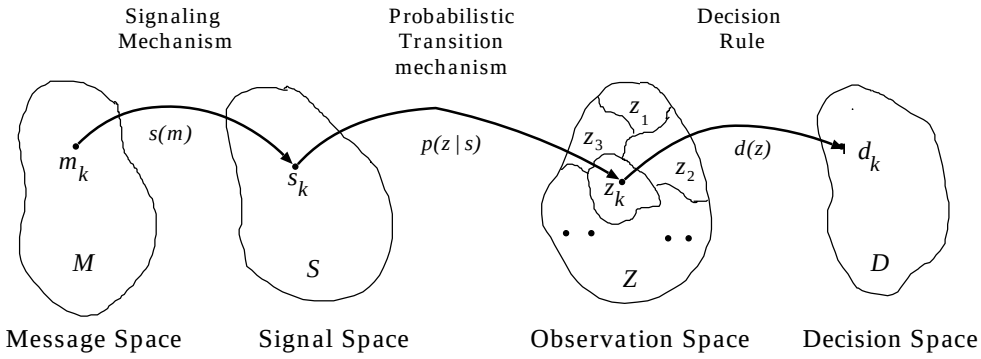


Figure 2.1: Structure of Decision Problems.

one of estimation. Often, but not always, there may be a probability measure associated with this space which tells us the way in which the messages are selected. The concept of signal space is used to isolate the portion of the problem where information is generated from the portion where it is transmitted. It is assumed that there is a unique and invertible mapping between the elements of the message and the signal space. Frequently, a model will be assumed for the physical mechanism that relates the observation space to the signal space. In the context of communications, this could be a channel model. Finally, there is a lot of flexibility in the decision space, D. The relationship between the observation space (which could be continuous or discrete), Z, and decision space, D, is called the decision rule. These four spaces and their interrelations are depicted schematically in Figure 2.1. The problem considered in this dissertation is explained in terms of these spaces at the end of this section.

The signal space $S = \{s_1, s_2, \ldots, s_K\}$ contains K elements and the decision space consists of J elements: $d_1, d_2, \ldots, d_J$. Quite often, $J = K$ and there is a logical pairing of messages and decisions, that is, if m_i is sent then d_i is the correct decision. This pairing, however, is not necessary. Two of the common decision criteria for multiple decision problems (more than 2 points in each of the four spaces) are the probability of error and the Bayes risk. In the Bayes risk criterion, each of the possible decisions is assigned a cost and then the overall average cost is minimized. So, associated with

the pair of message m_k and decision d_j, is a cost C_{jk} defined as the cost of deciding d_j given that m_k was sent. Also, it is assumed that the a priori probabilities, $\Pr(m_k)$, of each message m_k are known.

In practice, finding the solution to the Bayes risk problem is difficult. However, when the probability-of-error cost is assumed, there is considerable simplification of the decision rule. This special case is, in fact, the probability-of-error decision rule. Here, we assume that the number of messages is the same as the number of decisions and that there is a logical pairing of message m_i and decision d_i. The cost matrix for a minimum probability-of-error decision rule is given by:

$$C_{jk} = \begin{cases} 0 & \text{if } j = k \\ 1 & \text{if } j \neq k \end{cases} . \tag{2.1}$$

For a general cost function, the average cost is given by:

$$\sum_{j=1}^{J} \sum_{k=1}^{K} C_{jk} \Pr(d_j, m_k) , \tag{2.2}$$

where $\Pr(d_j, m_k)$ is the probability that the decision d_j was taken when the message m_k was sent. Since $\Pr(d_j, m_k) = \Pr(d_j|m_k) \Pr(m_k)$, the average cost becomes

$$\sum_{j=1}^{J} \sum_{k=1}^{K} C_{jk} \Pr(d_j|m_k) \Pr(m_k) . \tag{2.3}$$

The observation space Z is partitioned into J sets $Z_1, Z_2, \ldots, Z_J$ and if the observation $z \in Z_j$ the decision d_j is made. Hence,

$$\Pr(d_j|m_k) = \sum_{z \in Z_j} \Pr(z|m_k) .$$

The average cost function Eqn 2.3 can be minimized by selecting Z_j, $j = 1, 2, \ldots, J$, such that $z \in Z_j$ if

$$\sum_{k=1}^{K} C_{jk} \Pr(z|m_k) \Pr(m_k) \; < \sum_{k=1}^{K} C_{lk} \Pr(z|m_k) \Pr(m_k) , \tag{2.4}$$

for all $l = \{1, 2, \cdots, j-1, j+1, \cdots, J\}$ (Melsa and Cohn 1978). Substituting the cost function of Eqn 2.1 into Eqn 2.3, we find that $z \in Z_j$, if

$$\sum_{k=1, k \neq j}^{K} \Pr(z|m_k) \Pr(m_k) \; < \sum_{k=1, k \neq l}^{K} \Pr(z|m_k) \Pr(m_k) . \tag{2.5}$$

In case of a tie, z is randomly assigned to one of the tied regions. We note that the summations on both sides of Eqn 2.5 are identical expect that a different term is missing in each. Adding and subtracting $\Pr(z|m_j)\Pr(m_j)$ on the left hand side of Eqn 2.5 and adding and subtracting $\Pr(z|m_l)\Pr(m_l)$ on the right hand side we have,

$$\sum_{k=1}^{K} \Pr(z|m_k)\Pr(m_k) - \Pr(z|m_j)\Pr(m_j) \; < \sum_{k=1}^{K} \Pr(z|m_k)\Pr(m_k) - \Pr(z|m_l)\Pr(m_l) \; . \tag{2.6}$$

Now, canceling the common term and changing signs, we have the following definition of the decision region Z_j:

$$\Pr(z|m_j)\Pr(m_j) \; > \Pr(z|m_l)\Pr(m_l) \tag{2.7}$$

for all $j = \{1, 2, \cdots, l-1, l+1, \cdots, J\}$. Hence, Eqn 2.7 indicates that we must compute $\Pr(z|m_k)\Pr(m_k)$ for $k = 1, 2, \ldots, K$ and then select the decision corresponding to the value of k for which $\Pr(z|m_k)\Pr(m_k)$ is maximum. In other words, $z \in Z_j$ if

$$\Pr(z|m_j)\Pr(m_j) = \max_k \Pr(z|m_k)\Pr(m_k) \; . \tag{2.8}$$

If we divide both sides of the above equation by $\Pr(z)$, then we find that $z \in Z_j$ if

$$\frac{\Pr(z|m_j)\Pr(m_j)}{\Pr(z)} = \max_k \frac{\Pr(z|m_k)\Pr(m_k)}{\Pr(z)} \; . \tag{2.9}$$

Using Bayes rule, the above means that the decision region Z_j is given by the values of z for which $\Pr(m_j|z) = \max_k \Pr(m_k|z)$. In other words we select decision d_j if the message m_j has the maximum a posteriori probability (MAP). This shows that for a multiple decision problem the minimum probability-of-error rule is equivalent to the MAP rule. A more detailed treatment of decision and estimation theory can be found in text books like (Melsa and Cohn 1978) and (Poor 1988).

In this dissertation we will be designing a MAP decoder for variable-length encoded sources transmitted over noisy channels. In our case, all the four spaces considered are discrete. The message space $\mathcal{M}$ consists of a finite collection of *sequences* of real numbers (drawn from a finite set of real numbers) which represent a quantized source sequence. Let the cardinality of $\mathcal{M}$ be K. In our problem, the message is encoded using a variable-length binary code which means that there is a one-to-one mapping between the set of quantized values and the set of binary codewords. This also implies

that there is a one-to-one mapping between the set of all *sequences* of real numbers to the set of all *sequences* of binary codewords. Now, the signal space, $\mathcal{S}$, is this set of all possible binary *codeword sequences*. Hence, the cardinality of $\mathcal{S}$ is also K. Since the codewords have varying lengths (in bits), the different points in $\mathcal{S}$ will typically have different number of bits. If $\mathcal{B}$ is the set of all possible lengths (in bits) of the sequences, then, the observation space consists of all possible $B \in \mathcal{B}$ bit sequences. Note that the number of points in the observation space is more than the number of points in the other three spaces, since there may be sequences of bits that do not correspond to any code sequence. Hence, the problem in this dissertation, is to find the partition of the observation space into K regions that minimizes the probability of making an erroneous decision. Note that in our problem, the signal space and the decision space are in fact the same.

2.2 Previous Work in Joint Source-Channel Coding

Joint source-channel coding (JSCC) schemes have been in existence since the early 1960's. Through the years these schemes have been introduced in many applications and can be broadly classified into three main categories: *joint source-channel encoders* (JSCE), which are techniques of designing the encoder part of the system in some joint sense; *joint source-channel decoders* (JSCD), which focus on the decoder; and methods that perform rate allocation between the source and channel codes based on some criterion. These categories are shown in Figure 2.2. In the last category the source and the channel coders are operated in tandem, but the bit allocation between the source and the channel coders is performed so as to minimize the overall distortion in the system. Examples of such work include (Modestino, Daut, and Vickers 1981) and (Hochwald and Zeger 1997).

The JSCE techniques can be classified further into three different branches. In the first, called *integrated* source-channel encoders, the source and channel coding is done by a single coder. One example of such work can be found in Dunham and Gray (1981), where the existence of joint source-channel trellis coders was proved. It has also been recognized that some bits in a stream are more sensitive to errors than others. Techniques that protect the different parts of the signal to different degrees form the class of *unequal error protection* JSCEs (Ho and Kahn 1996; Zeger

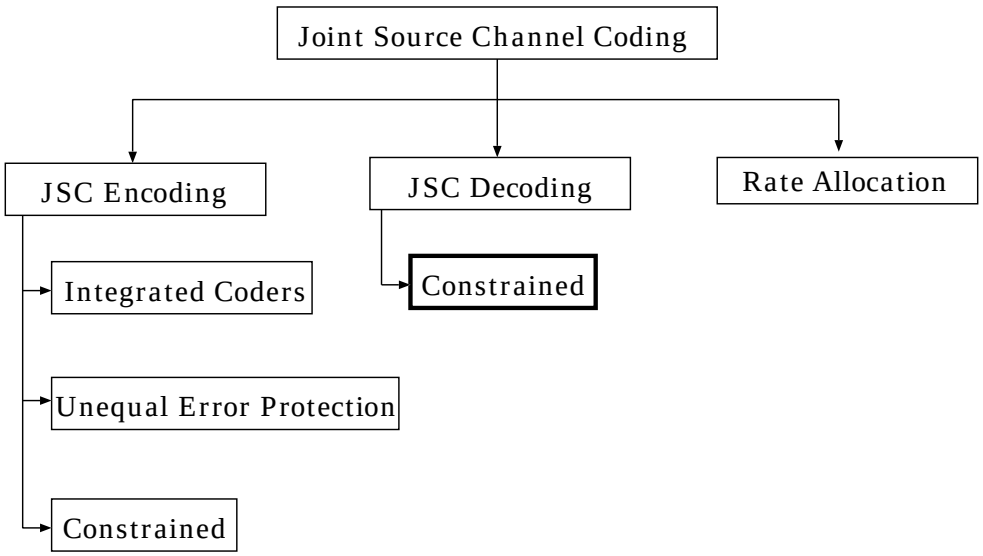


Figure 2.2: Classification of Joint Source-Channel Coding Methods

and Gersho 1990; Rydbeck and Sundberg 1976). The third kind of JSCE is the *constrained* source-channel coding class, where the source coder is designed under a suitable constraint. For example, Farvardin (1990) proposed a vector-quantizer design algorithm that uses the knowledge of the BSC-channel cross-over probability to minimize the mean squared-error (MSE). This work was then extended to tree-structured and multi-stage vector quantizers by Phamdo et al. (1993). Quantization of memoryless and Gauss-Markov sources for binary-Markov channels was proposed in Phamdo, Alajaji and Farvardin (1997).

Constrained source-channel decoders are built using prior knowledge of channel characteristics and there are two main types: those decoders that attempt to maximize a probabilistic metric based on the source statistics, the channel statistics and the observed sequence; and those that minimize the MSE of the reconstructed sequence. Examples of the first type can be found in (Sayood and Borkenhagen 1991) and this work has been extended using a maximum a posteriori (MAP) decoder (Phamdo and Farvardin 1994). A MAP decoder based on hidden Markov models was developed by

Park and Miller (1997). Representative work of the second, MSE, category can be found in (Park and Miller 1998b).

In this dissertation we propose new joint source-channel decoders for entropy coded sources over both channels with and without memory. These decoders use the source statistics and the channel statistics to improve the overall performance of the codec. Hence, these decoders belong to the class of constrained joint source-channel decoders which is indicated by a thicker box in the Fig 2.2. We note that JSCDs, broadly speaking, can be used along with JSCE techniques in order to increase the overall performance of the codec. The JSCD technique proposed in this dissertation could, potentially, be used with the constrained JSCE approaches mentioned in this section. Rate allocation techniques are also orthogonal to the JSCD presented in this dissertation and hence could be used in conjunction with them in order to increase the overall performance.

All the techniques mentioned so far (other than those presented in this dissertation) deal with fixed length encoded systems. The main reason for this, is that dealing with variable codes under noisy conditions becomes especially tricky. Some of the effects of channel errors on variable length encoded systems will be discussed in the following section. To the best of the author's knowledge, there are two other groups that have been concurrently working on designing MAP decoders for variable length codes. Their work can be seen in (Park and Miller 1998a) and (Demir and Sayood 1998). The MAP-BSC decoder proposed in this dissertation differs from the above mentioned work in that it does not assume the number of bits or the number of symbols transmitted and proposes a novel state-space structure to control the computational-complexity. Consequent to their work in (1998a), Park and Miller (1999) present suboptimal decoders that deal with the complexity issue by not requiring that the actual MAP sequence be found. Our approach finds the true MAP sequence in a computationally efficient manner. In this dissertation we propose a MAP decoder for entropy coded sources transmitted over a binary *Markov* channel for the first time. This dissertation also applies the MAP-BSC decoder to an image coder based on subband decomposition for the first time.

2.3 Effects of Channel Errors on Variable-Length Encoded Signals

Transmission errors can be classified roughly into random bit errors and erasure errors. While random bit errors cause single bit inversions, deletions or additions, erasure errors affect a series of bits. Examples of erasure errors are those due to packet loss in packet networks and burst errors. In a system that uses only fixed length codes (FLC), the effect of a random bit inversion is localized to the single word in which this error was actually introduced. However, when variable length codes (VLC) are used, a random bit inversion could cause more than a single codeword to be affected. There are three possible outcomes of a single bit inversion in a VLC encoded bit stream. In the simplest case, a codeword is mapped to another codeword of the same dimension, in which case there is no synchronization loss and the damage is limited to one codeword only. In the second case, part of a codeword may be decoded as another of smaller dimension or the current codeword may combine with parts of another to form a codeword of a larger dimension. In that case the decoder and the encoder are out of synchronization, and the decoder is unaware of it. Lastly, the codeword may be mapped to a word of the same dimension, but *not* in the codebook and the decoder stops decoding. In such cases some strategy is used that will allow the decoder to continue decoding, perhaps with the loss of a few bits.

To illustrate loss of synchronization, we consider a simple variable length codebook given by: $A : 0$; $B : 10$; $C : 110$; $D : 111$. Let us assume that the transmitted sequence was 010110 corresponding to the sequence of codewords A, B, C. Let the first bit be received in error, making the received sequence 110110. The receiver will interpret this as the sequence C, C. Hence we end up with two codewords in place of the three that were actually transmitted. Here, the synchronization is lost at the first bit and regained at the fourth bit. The synchronization is said to be regained because the word boundaries match up again. In some cases, even after synchronization is regained, the following data may be rendered useless since the temporal or spatial locality information of the decoded data is lost.

2.3.1 Tackling Synchronization Loss

In this section, we briefly look at some of the various methods that have been proposed in the literature to tackle the synchronization loss problem. One among the first works in this area was that of Ferguson and Rabinowitz (1984), where they showed that there are often substantial differences in the resynchronization properties of the different Huffman codes that could be designed for the same source and developed a method of constructing self-synchronizing Huffman codes. Self-synchronizing Huffman codes are those where the Huffman codebook contains one or more codewords that resynchronizes the decoder and encoder regardless of previous slippage. However, they also showed that self-synchronizing codes cannot be constructed for all source distributions and developed some sufficient conditions for the existence of self-synchronization codes for some sources. Extended synchronization codes were later proposed for binary prefix codes (Lai and Kulkarni 1996), which basically incorporated an extra synchronizing codeword into the design algorithm. Titchener developed a whole new set of entropy codes called the T-Codes (Titchener 1997) that have better resynchronization properties. A class of parameterized, reversible variable-length codes were proposed by Wen and Villasenor (1997), which can be decoded from either end of the transmitted stream giving rise to a significant increase in error-resilience. All these techniques can be used in conjunction with the proposed decoders since they are, essentially, special types of variable-length codes. Another technique which can be clubbed with almost any other approach is to use end-of-line (EOL) markers to help reduce the effect of synchronization loss in variable-length encoded systems. In this approach a special symbol (highly protected using forward error correcting codes) is transmitted periodically and the correct receipt of this symbol lets the decoder regain synchronization with the encoder. EOL symbols are often used in image transmission since images are much more sensitive to loss of synchronization than 1-D signals like speech. This is so because, in speech like signals the loss of synchronization results in a few pops and clicks and once the synchronization is regained the rest of the speech is clearly audible. On the other hand, in images, the loss of synchronization results in extra pixels to be added or a few pixels to be deleted in a single row and the spatial locality information of the subsequent pixels is lost which causes distortion of the image.

Redmill and Kingsbury (1996) developed an error resilient entropy coding (EREC) technique which approaches the problem from an entirely different angle. Their technique is based on reordering the variable length bit stream such that each variable-length block starts at a known position. While this technique can avert synchronization loss, no word errors are corrected.

All the above approaches deal with the problem of synchronization loss at the encoder end and do not fall under joint source-channel coding schemes. In some cases, redundancy is introduced for better resynchronization. In this dissertation we propose a joint source-channel coding algorithm that approaches the problem from the other end, namely the decoder. This technique will use the redundancy already present in the source to limit the damage due to channel errors on variable-length encoded sources. Most of the methods mentioned above can be used along with the JSCD presented in this dissertation.

2.4 Performance Measures Used in this Dissertation

The performance of the MAP decoder is evaluated using two parameters: the percentage of bits that are out of synchronization (PBOS) and the modified signal to noise ratio (MSNR) of the decoded stream. The MSNR is used in lieu of the regular SNR for 1-D signals because it does not penalize the decoder too severely for loss of synchronization and is suited for speech like signals, where the synchronization loss is heard merely as short bursts of garbled speech. It must be noted that the definition of MSNR is generally application dependent. For signals like images, where spatial locality information is very crucial perceptually, we use the peak signal-to-noise ratio (PSNR) as a measure of the performance of the decoder. The PBOS is calculated based on the average number of bits that are out of synchronization over all stretches of synchronization loss. This performance metric measures the efficacy of the decoder in controlling synchronization losses.

The MSNR is calculated as the SNR between the transmitted sequence and a sequence synthesized from the decoded sequence. In Figure 2.3, the streams marked "Tx","Rx" and "Synth", represent the transmitted, decoded and synthesized se-

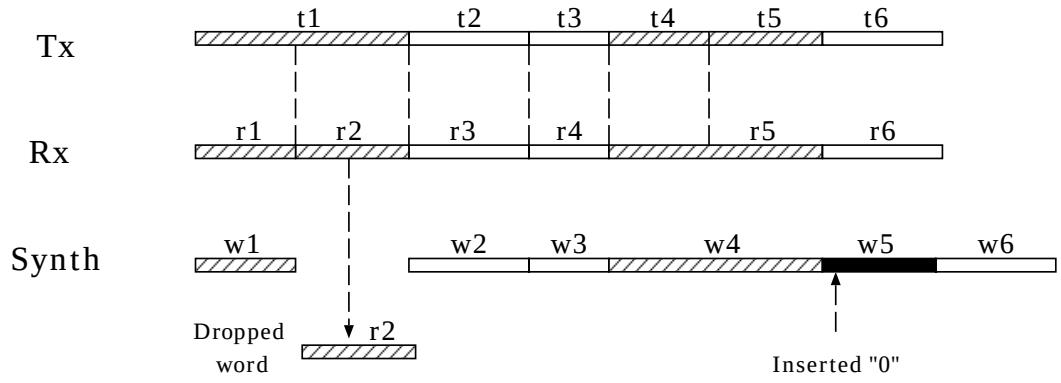


Figure 2.3: Calculating MSNR

quences respectively. The vertical lines in the streams represent the word boundaries. Upon aligning the two streams bitwise, it is seen that there are two asynchronous portions in the stream separated by two synchronous portions. The shaded area in each stream represents the asynchronous portion of the stream and the clear area represents the synchronous portion. In the first asynchronous portion one transmitted word (t1) is decoded as two words (r1 and r2) and in the second asynchronous portion two transmitted words (t4 and t5) are merged into one decoded word (r5). The synthesized stream is created such that the words in the synchronous portion of the decoded stream remain at the same word position as the corresponding words in the transmitted sequence. So, if the asynchronous portion of the decoded stream consists of more words than the corresponding portion of the transmitted stream, we truncate the decoded stream and when the decoded portion has smaller number of words, we pad it with zeros. In this example, the first asynchronous portion of the decoded stream has one more word than the corresponding portion of the transmitted stream and hence the word r2 is dropped from it. In the second asynchronous portion, the decoded stream has one word less than the transmitted stream and a zero word, represented by a black block, is inserted.

As noted before, this method of calculating the MSNR reduces the penalty for synchronization losses and is suited for speech like signals, where the synchronization

loss is heard merely as short bursts of garbled speech. It must be noted that the definition of MSNR is generally application dependent. For signals like images, as mentioned before, we use the PSNR which is calculated as follows

$$\text{PSNR} = 10 \log_{10} \left(\frac{255 \times 255}{\sum_i (x_i - \hat{x}_i)^2)} \right) , \tag{2.10}$$

where x_i represents the original image pixel value at position i and $\hat{x}_i$ represented the reconstructed image pixel value at position i. Note that while calculating the PSNR, we match the original and reconstructed images directly unlike in the MSNR case.

2.5 Signal Compression

In this dissertation, the MAP decoders are tested for different sources over different codecs. The basic building blocks of these codecs are studied in this section, in the context of a generic signal compression system shown in Fig. 2.4. The model shown in Fig 2.4 has three modules in the transmitter: the transform, the quantizer and the entropy coder. The following sections discuss these components in greater detail. While the coding scheme used to test the one dimensional MAP decoders to be developed in Chapters 3 and 4, use scalar quantizers (Section 2.5.2) and Huffman codes (Section 2.5.3); the image coding scheme developed in Chapter 5 uses almost all the components of the generic system described below.

2.5.1 Transforms

This section describes the first step (cf. Figure 2.4) in most signal compression schemes, namely, transforming the signal. The aim of this process is to transform the signal into a more compressible form. Such transformations can be considered in two different perspectives: the signal processing and the linear algebraic. When the signal is decomposed into different frequency bands using a set of filters it is called the *sub-band decomposition* of the signal. When the signal is operated on by a transform (like the discrete cosine transform or the Fourier transform) then the associated coding method is called transform coding. Transform coding can be considered as a special case of sub-band coding (Vaidyanathan 1993).

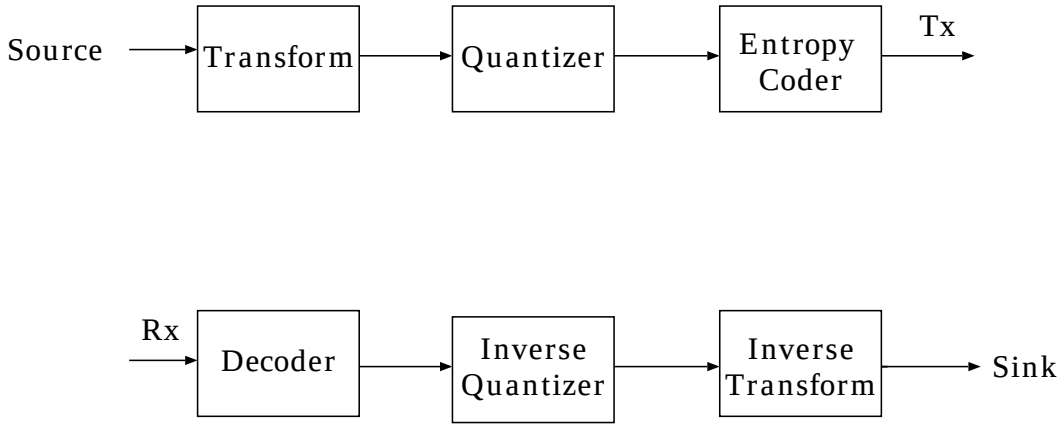


Figure 2.4: A Generic Signal Compression Scheme

Transform Coding

Transform coding is a method in which the image is decomposed into several components that are then coded according to their specific characteristics. The idea behind this method is to compact the information in the image into a few elements (called coefficients) that are easier to code. A simple way to illustrate this method would be to consider the data scatter plot shown in Fig 2.5 where two actual data points are taken together to form a 2-D vector describing a single point in the 2-D plane. Now it can be seen that the data is mostly concentrated around the α axis when we use the α-β co-ordinate system. So, if the points are resolved along these axes, then their spread is smaller along one of the axes (the β axis). The "information" contained in two values of the original data now sequence gets more concentrated in one of the values and the change in the variance of the signal along the two directions leads to better compression (Gersho and Gray 1992). Note that the transform itself does not compress the signal, it merely expresses the signal in a form that is more conducive to compression.

Mathematically, a transform projects the signal onto a set of basis functions. If $\mathbf{x} = (x_1, x_2 \cdots, x_k)$ is a k-dimensional vector in the original space and $\mathbf{T}$ is a $k \times k$

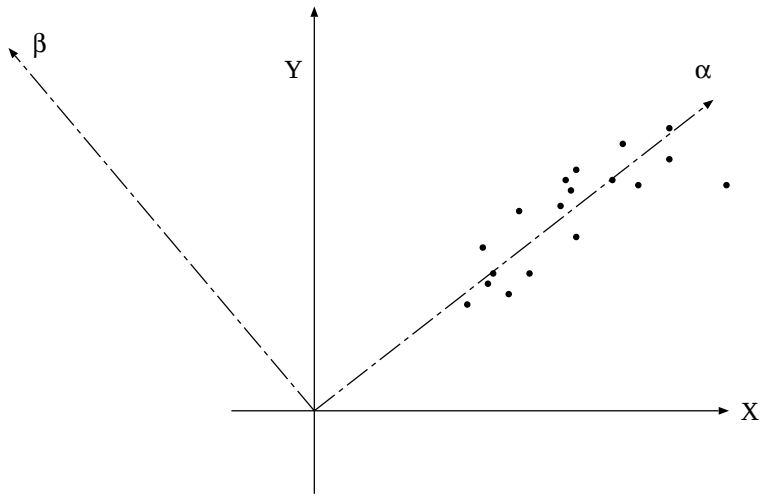


Figure 2.5: Transform Coding: A geometric perspective

matrix of basis vectors, then the projection $\mathbf{y} = (y_1, y_2 \cdots, y_k)$, of $\mathbf{x}$ onto the basis functions is given by:

$$\mathbf{y} = \mathbf{T}\mathbf{x} \ . \tag{2.11}$$

Note, that for a lapped orthogonal transform (Malvar and Staelin 1989), the transform matrix is $k \times l$, with $l \geq k$. This means that the input is partitioned into overlapping blocks of length l and each block transformed into a block of smaller length k. Lapped orthogonal transforms have been shown to reduce blocking effects in image and speech signal processing. The inverse transformation can be obtained by multiplying the vector $\mathbf{y}$, by the inverse of the matrix $\mathbf{T}$, $\mathbf{T^{-1}}$:

$$\mathbf{x} = \mathbf{T^{-1}}\mathbf{y} \ . \tag{2.12}$$

If the transformation is unitary, then $\mathbf{T}^{*T}\mathbf{T} = \mathbf{I}$, where $\mathbf{T}^{*T}$ is the transpose of the complex conjugate of the matrix $\mathbf{T}$ and $\mathbf{I}$ is the identity matrix. This property is useful in coding, because this means that the quantization error magnitude in the transform domain is the same as that in the signal domain and the quantizer can be optimized in the transform domain. When the signal is two dimensional (like in

an image), the inputs and outputs of the transform coding system are described as matrices. Hence, now the forward transform equation becomes

$$\mathbf{Y} = \mathbf{T}^T\mathbf{X}\mathbf{T} \ , \tag{2.13}$$

where $\mathbf{X}$ is an input matrix and $\mathbf{Y}$ the corresponding transformed matrix. Usually, separable transforms are used for image coding applications. This allows the 2-D transforms to be split into two 1-D transforms, which can then be applied separately along the horizontal and vertical directions.

A transform can also be viewed in terms of the changes in statistics between the original and transformed sequences (Sayood 1996). In this perspective, a transform helps to decorrelate the samples of the signal. When the correlation between the samples are reduced, better signal compaction is possible. One such transform was proposed by Karhunen and Loève and is known as the Karhunen-Loève transform or the KLT (Sayood 1996). The KLT is tailored to the particular signal that is being quantized and is optimal in that sense. Other popular transforms include the discrete cosine transform (DCT) and the class of wavelet transforms. Detailed descriptions of various transforms and their application to signal compression can be found in textbooks like (Sayood 1996; Rao and Yip 1990), and (Gersho and Gray 1992). In our test of the MAP-BSC algorithm for images, we will be using a wavelet transform scheme to decorrelate the signal (cf. Section 5.2.6).

Subband Coding

Subband coding (Woods 1991), as mentioned earlier, is based on decomposing a signal into a set of frequency subbands by using a filter called the analysis filter. Separating the signal into different frequency components and treating them separately can yield better compression performance (as discussed in the case of transform coding). The outputs of the analysis filters (h_0 and h_1, for a special case of 2 channels) are sub-sampled to maintain critical sampling and the resulting coefficients are transmitted to the receiver. At the decoder end, the received coefficients are interpolated and a set of filters, called the synthesis filters (g_0 and g_1), operate on the received coefficients to reconstruct the signal. A block diagram representing the above scenario is shown in Fig 2.6. The analysis and synthesis filters are chosen so that aliasing effects due to sub-sampling cancel out. With proper filter selection the above system is called the

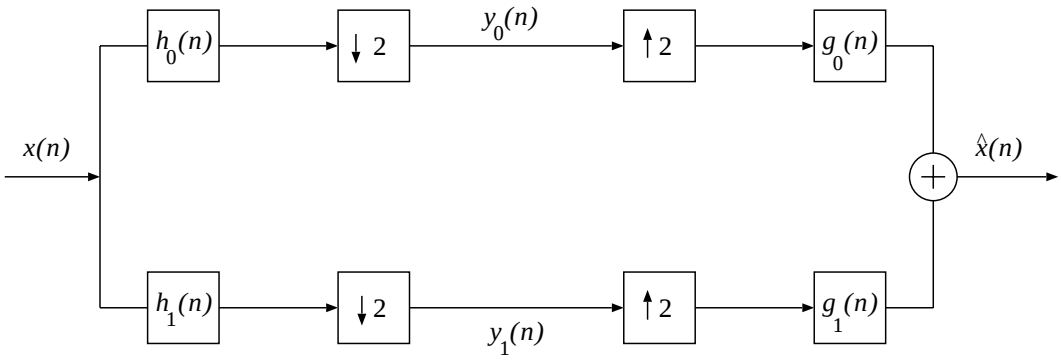


Figure 2.6: A 2-Channel Perfect Reconstruction Filter Bank

two-channel perfect reconstruction (PR) filter bank. If there is no quantization of the signals at the encoder and if there are no errors in the channel through which the signal is transmitted, then the reconstructed signal is an exact replica of the transmitted signal. If $X(z)$ is the representation of the signal $x(n)$ in the z-transform domain, $H_0(z)$, $H_1(z)$, $G_0(z)$ and $G_1(z)$ are the frequency responses (in the z-transform domain) of the analysis low-pass, high-pass and synthesis low-pass and high-pass filters respectively and $\hat{X}(z)$ corresponds to the reconstructed signal, $\hat{x}(n)$, then we have (Vetterli and Kovacevic 1995)

$$\hat{X}(z) = \frac{1}{2}[H_0(z)G_0(z) + H_1(z)G_1(z)]X(z) + \frac{1}{2}[H_0(-z)G_0(z) + H_1(-z)G_1(z)]X(-z) \ . \tag{2.14}$$

For perfect reconstruction,

$$\hat{X}(z) = X(z) \ , \tag{2.15}$$

which implies that the coefficients of $X(z)$ must add up to 1; or

$$H_0(z)G_0(z) + H_1(z)G_1(z) = 2 \ . \tag{2.16}$$

Also, all the aliasing terms which contain the factor $X(-z)$ must cancel,

$$H_0(-z)G_0(z) + H_1(-z)G_1(z) = 0 \ . \tag{2.17}$$

In practice it is desirable to deal with finite impulse response (FIR) filters. In order to achieve PR with FIR filters, it is required that

$$H_0(z)H_1(-z) - H_1(z)H_0(-z) = 2z^{-2k-1} \ , \tag{2.18}$$

where k is any integer (Vetterli and Kovacevic 1995). Substituting Eqn 2.18 into Eqns 2.16 and 2.17, we have the definition of the synthesis filters in terms of the analysis filters:

$$G_0(z) = z^{2k+1}H_1(-z) \tag{2.19}$$

and

$$G_1(z) = -z^{2k+1}H_0(-z) \ . \tag{2.20}$$

From the above equations it is clear that the synthesis filters will be non-causal if the analysis filters are causal. The synthesis filters can be made causal by multiplying by z^{-2k-1}, so that Eqn 2.15 becomes

$$\hat{X}(z) = z^{-2k-1}X(z) \ , \tag{2.21}$$

describing perfect reconstruction with delay $2k+1$. Extensive research has been done to solve the Eqns 2.16 and 2.17 (Vaidyanathan 1993; Vaidyanathan 1987).

Often, multichannel filter banks are constructed from the simple two-channel filter bank system described above. In this case, the two-channel filter bank system is cascaded appropriately. An example is shown in Fig 2.7, where the frequency analysis is obtained by iterating the two-channel division on the previous low-pass channel. This is called an *octave-band* filter bank since each high-pass output contains an octave of the input bandwidth. It is also possible to apply the two-channel splitting to both the high-pass and the low-pass channels resulting in a tree with 2^K leaves, where K is the number of levels in the expansion. An arbitrary tree-structured filter bank may also be built, where only some of the channels are split further. For a two dimensional source and separable filters, there are two sets of analysis and synthesis filters : one for the vertical direction and the other for the horizontal direction. An example of an octave-band filtered image (for three stages of filtering) is shown in Figure 5.4. For a more detailed discussion of octave-band filtering in the context of images, the reader is referred to Section 5.2.2.

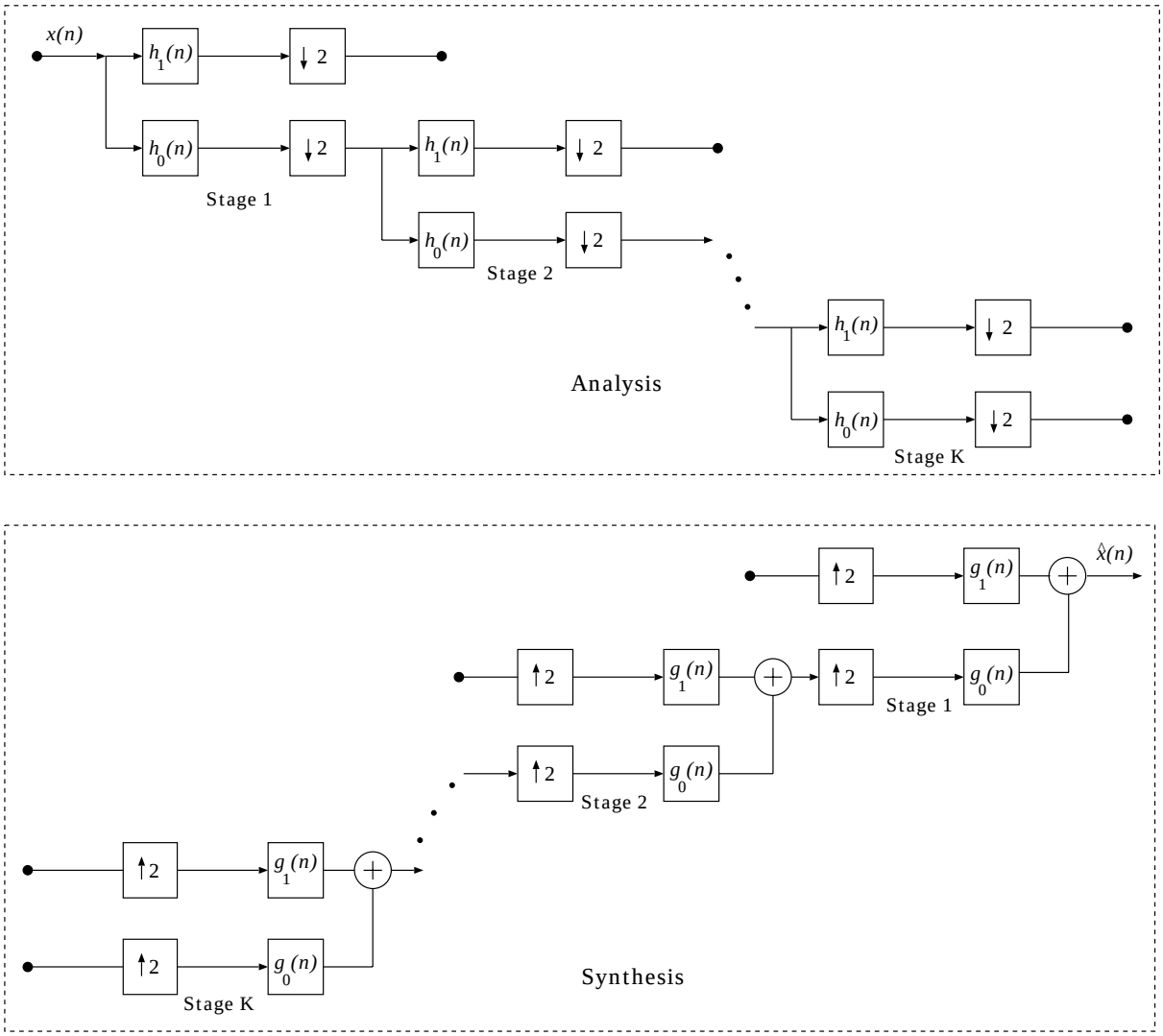


Figure 2.7: An Octave-band Filter Bank with K stages

Wavelet Decomposition

With the discovery that continuous time wavelets can be generated from discrete-time filters and vice-versa (Mallat 1989; Daubechies 1988), the design of wavelets and filter banks have become connected. Wavelets can be used to generate filters that can be used for subband coding. The mathematics behind wavelets is involved and a detailed exposition is beyond the scope of this dissertation. There are many books covering wavelets, which include (Burrus, Gopinath, and Guo 1998; Vetterli and Kovacevic 1995) and (Strang and Nguyen 1997). A mathematically more rigorous treatment of wavelets can be found in Daubechies (1992).

Broadly, one might classify the wavelet basis functions into orthonormal and biorthogonal bases. In the case of orthonormal wavelets, the basis functions are orthogonal to each other. The most important orthonormal wavelets for image coding, are those that are compactly supported. A systematic way of generating compactly supported orthonormal wavelets was proposed by Daubechies (1988) and a fast algorithm for computing a discrete wavelet transform (DWT) was given by Mallat (1989). A major disadvantage of compactly supported orthogonal wavelets, is their asymmetry, which translates into nonlinear phase in the associated finite impulse response (FIR) filters. The problem with nonlinear phase filters is that the group delay associated with the filter is not a constant. This may cause unpleasant distortions that may not be present when the phase is linear when the signal is quantized.

There are two ways of obtaining symmetry in wavelets: one is to relax the orthogonality condition and the other is to give up on compact support (except for the Haar wavelet which is orthogonal and compactly supported). Giving up compact support leads to additional computational complexity which is not desirable and hence biorthogonal filters are often used in image compression. Biorthogonal wavelets are those in which the basis functions are not orthogonal to each other; however, they are orthogonal to another pair that is used to compute the inverse DWT. The perfect reconstruction property is preserved and the fast algorithm proposed by Mallat can still be used. The fact that different bases can be used for the analysis than for synthesis implies that there is more flexibility in a biorthogonal system than in an orthonormal system. Note, that the energy of the signal in the transform domain need no longer be equal to that in the time domain when using a biorthogonal wavelet; however, the

difference is quite small for "useful" filter sets.

2.5.2 Quantizers

Compression algorithms can be classified into two types: *lossless* and *lossy*. When the reconstructed signal is *exactly* the same as the original signal, then the compression technique is called lossless; whereas, if the reconstructed version is allowed to be different from the original signal, then the compression technique is said to be lossy. In this section, we will see one example of lossy compression, viz., quantization.

The term quantization can refer to the process by which a discrete time, continuous amplitude signal is converted into a discrete time, discrete amplitude (digital) signal or to the process of providing a coarser description of a digital signal that requires fewer bits than the original. In this dissertation, we will deal with quantization as a way of reducing the number of bits required to describe a signal.

The transformation of the signal, as described in the previous subsection, aids in concentrating the energy of the signal into fewer number of components from among those into which the signal is resolved. The decomposed signal can then be passed through a quantizer designed for it, which leads to compression of the signal. In the next two subsections we will describe scalar and vector quantizers (VQ). Scalar quantizers will be used in the coder designed to test the proposed decoders in Chapter 3, 4 and 5, whereas, VQs will be used in Chapter 5 only.

Scalar Quantization

In its simplest form a scalar quantizer (SQ) observes a single source symbol and selects the nearest approximating value from a set of predefined finite set of (say N) allowed values (Gersho and Gray 1992). Associated with an N-level scalar quantizer is a partition of the real-line into N cells and a representation value for each cell. The characteristics of the scalar quantizer is then determined by these two sets of values. The cells at the either end of the real-line are unbounded on one side each and are called the overload cells. A SQ can be classified into *uniform* or *non-uniform* based on whether the cells (other than the overload cells) are of equal size or not. The quantizer can also be classified as *mid-tread* or *mid-rise* depending on whether the origin of the real-line is contained in a bin or is at the boundary of a couple of

bins, respectively. A rigorous treatment of scalar quantizers can be found in Gersho and Gray (1992). In this dissertation we use uniform quantizers only, since they are not very complicated to design and have been shown to perform almost as well as more complex, source-pdf matched quantizers for a class of non-Gaussian memoryless sources (Farvardin and Modestino 1984). Also, most image and video compression standards use uniform quantizers rather than non-uniform quantizers when used in conjunction with entropy codes.

Vector Quantization

Vector quantization is a popular source coding technique and is a generalization of the scalar quantization to the quantization of a vector, in which k source symbols are grouped together into a k-dimensional vector and then quantized. This can be especially useful, when the samples are statistically dependent. When the dependence between the samples is *not* linear, linear transformations cannot completely remove the dependence form the signal. Under these circumstances, a vector quantizer (VQ) can be used to take care of the dependence between the samples. However, a VQ can give superior performance over scalar quantizers even when the components of the signal are statistically independent of each other, since a VQ offers more flexibility in the partitioning of the input space, than a scalar quantizer. All the above reasons motivate the study of vector quantization.

Vector quantization is a pattern recognition technique, where an input pattern is approximated by one of a set of predefined patterns. A VQ of dimension k and size N is a mapping from a vector in k-dimensional Euclidean space, $\mathcal{R}^k$, into a finite set, $\mathcal{C}$, of N output points called codewords. The set of all codewords is referred to as a codebook. The rate of the VQ is $r = \log_2(N)/k$, when no entropy coding is used. The rate can be further reduced when entropy codes are used, in which case a slightly different flavour of VQ is usually used to get the best out of the scheme, as will be described later.

Associated with every vector quantizer, is the codebook design algorithm and the encoding algorithm. The codebook design algorithms determines the partition of $\mathcal{R}^k$ into N regions (for a VQ of size N) and the codeword for each of the regions. The encoding algorithm finds the closest codeword to the given input vector from among all

the codewords in the codebook. The codebook design algorithm is iterative and uses a sample "training set" of data, which it partitions into regions. The centroids of these regions are the codewords. In a non-adaptive coding application, the VQ codebooks are designed beforehand at the encoder and a copy if it exists at (or is sent as header information to) the decoder. When a signal needs to be coded, the encoder finds the codeword that is "closest" to the source vector and transmits the corresponding codeword index (either entropy coded or otherwise). The decoder simply picks out the codeword corresponding to the index and writes it out as the reproduction of the transmitted signal.

The performance of the VQ relies heavily on the design of the codebook. A "bad" codebook can lead to poor performance of the entire system. The overall performance of the VQ can be assessed by either a statistical average of a suitable distortion measure or by a worst case analysis. It is usual to concentrate on the statistical performance of the VQ. If $d(X^k, Y^k)$ were to denote the distortion between the k-dimensional source vector X^k and its reproduction Y^k, then the average per-sample distortion of the source is given by

$$D = \frac{1}{k} E[d(X^k, Y^k)] \ . \tag{2.22}$$

The aim of the VQ design algorithm is to minimize this distortion. An optimal VQ, will then be that which partitions the signal space and finds a representation for each of the partitions in such a way as to minimize this distortion. It has been proved (Linde, Buzo, and Gray 1980) that an optimal VQ satisfies the *nearest-neighbour*(NN) and the *centroid* conditions for a squared error distortion measure. Let the encoder be defined by the partition of $\mathcal{R}^k$ into the cells $R_1, R_2, \cdots, R_N$ and let the decoder be completely specified by the codebook $\mathcal{C} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots \mathbf{y}_N\}$. Then, according to the NN condition, given a set of output levels $\mathcal{C}$, the optimal partition cells satisfy

$$R_i \subset \{\mathbf{x} : d(\mathbf{x}, \mathbf{y}_j); \forall j\} \ , \tag{2.23}$$

that is, an input vector $\mathbf{x}$ will be quantized to $\mathbf{y}_i$, if and only if it is farther away from all codewords other than $\mathbf{y}_i$. The centroid condition states that, for a given partition $R_i; \{i = 1, 2, \cdots, N\}$, the optimal code vectors satisfy

$$\mathbf{y}_i = \frac{1}{||R_i||} \sum_{j=1}^{||R_i||} \mathbf{x}_j \ . \tag{2.24}$$

The necessary conditions for optimality provide the basis for iteratively improving a given VQ. If the iteration continues to convergence, a good (hopefully close to optimal) quantizer can be found. The iteration begins with a VQ consisting of its codebook and the corresponding optimal (NN) partition and then finds the new codebook which is optimal for that partition. This codebook and the NN partition form a new VQ whose average distortion is no more (and usually less) than the original quantizer. Even though the individual steps involved in the VQ design are simple, achieving global optimality is quite difficult, because the rate-distortion function has many local optima and it is very easy to get trapped in one of the local optima. The VQ design algorithm described above, is generally referred to as the generalized Lloyd algorithm (GLA) (Linde, Buzo, and Gray 1980).

The problem now, is selection of a suitable initial codebook. The initial codebook greatly affects the performance of the final VQ, because the locally optimal point that the algorithm converges to, will depend on this. There are a variety of techniques that have been developed to design the initial codebook, a survey of which can be found in Gersho and Gray (1992). They can be broadly classified into five different classes: random coding, pruning, pairwise nearest-neighbour, product codes and splitting. In the random coding approach, a codebook containing N codewords is randomly picked from the training set based on the statistics of the source. One simple way to do this is to pick every K^{th} codeword form the training set. Pruning refers to the idea of starting with the training set and selectively eliminating training vectors as candidate code vectors until a final set of training vectors remains as codebook.

Equitz (1989) proposed a technique called the pairwise nearest neighbour (PNN) clustering algorithm. This algorithm starts with all the vectors in the training set and ends up with N vectors that need not necessarily be part of the training set. The partitioning process in this algorithm proceeds by finding the distortion between each pair of vectors. Then the two vectors having the smallest distortion are put together in a single cluster and represented by either vector. If we had L training vectors, we now have $L-1$ clusters. The algorithm proceeds by merging two clusters at each stage until only N clusters remain. Another way to choose an initial codebook would be to use a product codebook. For example, if a codebook needs to be defined for a k-dimensional VQ, with codebook size 2^{kR} for some integer R, then one can use the product of k scalar quantizers with 2^R words each. Finally, there is the splitting

method, in which the codebook is grown from a smaller one. The globally optimal single word codebook for any VQ, is the centroid of the complete training set. This codeword, say $\mathbf{y}_0$, in this can be "split" into two codewords, $\mathbf{y}_0 + \epsilon$ and $\mathbf{y}_0$, where ϵ is a vector of small Euclidean norm. This can be no worse than the initial codebook, since the initial codebook is contained in it. Now, the GLA can be run using this as the start-up codebook to get a good two word codebook. Then, the two codewords in this VQ can be split again, just as before, to produce an initial 4 word codebook and the GLA can be run again. In this way a VQ codebook of N dimension can be created.

The GLA however, is only guaranteed to converge to a *local* optimum as opposed to a global optimum. A complex cost function in general will have many local minima, where some are better than others and it is likely that the GLA will get caught in one of the local optima. By introducing randomness into each iteration of the GLA, it is possible to evade local minima, reduce the dependence of the VQ on the initial codebook and locate a solution that may even be the globally optimal solution for the given cost function. The family of optimization techniques called *stochastic relaxation* (SR) are characterized by the fact that each iteration of the search for the optimal solution involves the perturbing of the *state*, the set of variables of the cost function (e.g., the codebook) in a random fashion. An extensive discussion of SR algorithms for VQ design can be found in Zeger, Vaisey and Gersho (1992).

It is possible to achieve further compression within the VQ paradigm by entropy coding (cf. Section 2.5.3) the VQ indices before transmitting them. In such a situation, it is better to design a vector quantizer that works towards reducing the average distortion *subject to a constraint on the entropy.* This leads us to the design of entropy constrained vector quantizers (ECVQ). In this variant, the GLA is modified to minimize a Lagrangian functional that trades off distortion for rate (Chou, Lookabaugh, and Gray 1989). The search criterion also uses the Lagrangian metric rather than the distortion metric. As in Equation 2.22, let D denote the average distortion between each source letter and its reproduction. If $l(X^k)$ is the length (in bits) of the binary word representing the k-dimensional codeword for the source vector X^k, then

$$R = \frac{1}{k} E[l(X^k)] \tag{2.25}$$

is the average rate of the source. The operational rate-distortion function is the

set of rate and distortion pairs that can be achieved for a particular source using various quantizers. The convex hull of this set of points, traces out a curve that represents "optimal" points at which to operate. In this algorithm, the convex hull of the operational rate-distortion function is found by minimizing the functional

$$J_\lambda = E[d(X^k, Y^k)] + \lambda E[l(X^k)] \tag{2.26}$$

rather than D, where λ can be interpreted as the slope of the line supporting the convex hull. In order to generate the entire convex hull, it is necessary to repeat the minimization of J_λ for various λ's. Generating the entire convex hull will let us choose the optimal quantizer at any required rate or distortion. The algorithm may be implemented using the entropy of the source or the actual rate that the source will be operating at. This rate can be found by including the design of an entropy code within the optimization loop. Since our object in this chapter is to test the JSCD developed for *variable* length encoded sources in a practical situation, and since one achieves better compression by using an entropy code, we used ECVQ rather than plain VQs in our experimental setup in Chapter 5.

VQs have low decoder complexity compared to transform coders and the VQ design procedure, though more complex, can be done off line. VQs can offer better compression than scalar quantizers even without the use of entropy codes. Even though there are advantages to using VQs, there are a few disadvantages as well. For instance, it is practically impossible to pick a training set that truly represents a large class of signals. Vector quantization is also not very easily made adaptive since it requires that the statistics of the source be learnt. This can be a problem when the signal statistics changes too soon and there is not enough data to make an estimate. Also, vector quantization does not work well with entropy coding techniques such as runlength coding and scalar quantization has been found to give equivalent or superior performance in applications such as image coding and hence vector quantization has not found its way into the current generation of standards.

Bit Allocation

Oftentimes, a signal is decomposed into various sub-sources (e.g: subbands in a wavelet decomposition) so that they can be better compressed. It is often possible to quantize these sub-sources at different rates, since they have different "tolerances" to

distortion. Deciding the rates at which to code these sub-sources, or in other words, allocating different bits to these sub-sources, becomes an interesting problem in itself. Suppose we have a set of k sub-sources, an overall bit budget of B bits with b_i being the number of bits allocated to sub-source S_i, then the overall distortion of the signal D can be defined in terms of the overall bit allocation vector $\mathbf{b} = (b_1, b_2, \cdots, b_k)$ according to

$$D = D(\mathbf{b}) = \sum_{i=1}^{k} W_i(b_i) \ , \tag{2.27}$$

where, $W_i(b_i)$ denotes the distortion incurred by the sub-source S_i, when it is optimally quantized at b_i bit resolution. The bit-allocation problem is to determine the optimal values of $b_1, b_2, \cdots, b_k$, subject to a given quota B of available bits, so as to minimize $D(\mathbf{b})$.

There are various ways in which bit allocation can be done. Some of the methods involve an explicit optimization as stated above, while the others are more implicit. When the bit-allocation is implicit, there is no special algorithm that allocates bits in any optimal fashion, for example, in the SPIHT algorithm (Said and Pearlman 1996) the small magnitude coefficients are coarsely encoded which implies fewer bits. There are two basic ways in which explicit bit allocation can be done. The first is to find an algorithm that exactly minimizes Eqn 2.27 over all choices of $\mathbf{b}$. The second way is to use the high resolution quantization approximations in order to write explicit expressions for $W_i(b_i)$ and then minimize the overall distortion.

In the first approach, if $\mathbf{b}$ were constrained to have integer components, then there are only a finite number of vectors, $\mathbf{b}$, over which, theoretically, maximization can be done by an exhaustive search. Apart from the computational intractability of the problem, there is also the fact that often the expression for $W_i(b_i)$ is not exactly known. However, it is still possible to do an optimal allocation using the above approach, when a set of good, if not optimal, quantizers and their corresponding values of $W_i(b)$ are known for different (not necessarily integer), b. Examples of such algorithms can be found in Shoham and Gersho (1988), Westerink, Biemond and Boekee (1988) and Riskin (1991). Examples of algorithms of the second type can be found in Huang and Schultheiss (1963), where the high resolution approximation theory is used to derive an optimal solution that overlooks the practical requirement that $b_i \geq 0$.

The generalized Breiman-Friedman-Olshen-Stone (BFOS) algorithm (Riskin 1991) is one of the methods to achieve optimal bit-allocation between sources, given a pool of quantizers (hence rate-distortion pairs) for each. The problem is to distribute a given number (or fewer) of total bits to each of the sources, so as to minimize the quantization noise of the overall signal. The generalized BFOS algorithm is an extension of an algorithm for optimal pruning in tree-structured classification and regression, to coding. This algorithm works by allocating a maximum number of bits to each class and then deallocating or pruning the bits optimally until the desired overall rate is achieved. The algorithm is briefly described, for integer bits, in the following paragraph. In Chapter 5, the non-integer version of the algorithm is used. More details about the algorithm can be found in Riskin (1991).

Let there be M sub-sources among which the given quota of bits needs to be allocated. Let q be the maximum number of bits that can be allocated to any source. In what follows, the operational rate distortion function is called quantizer function. If the quantizer function is convex, then

$$\left|\frac{d_i(0)-d_i(1)}{r_i(0)-r_i(1)}\right| > \left|\frac{d_i(1)-d_i(2)}{r_i(1)-r_i(2)}\right| \cdots , \tag{2.28}$$

where $d_i(j)$ is the average distortion and $r_i(j)$ is the average rate with j bits allocated to the sub-source i. As the rate increases, the magnitude of the slope of the quantizer function decreases.

Let the bit allocation b be an $M-$tuple of nonnegative integers that determines the rate and distortion $R(b)$ and $D(b)$ of the code. Let B be the set of all possible bit allocations, then the operational rate-distortion function can be expressed as

$$\hat{D}_T(R_d) = \min_{b\in B}\{D(b)|R(b) \leq R_d\} . \tag{2.29}$$

It specifies the minimum average distortion for the desired rate R_d under the condition that the quantizers are based on an allowable bit allocation.

Let j_i bits be allocated to a sub-source i for $i = 1, 2, \cdots, M$ in a given bit allocation so that the sub-source i measures an average distortion $d_i(j_i)$. Let p_i be the probability that the sub-source i is selected (or weighting factor for that source). We can express the overall average distortion and rate, D and R, as

$$D = \sum_{i=1}^{M} p_i d_i(j_i) \tag{2.30}$$

and

$$R = \sum_{i=1}^{M} p_i j_i \tag{2.31}$$

Assume that the next allocation is got by pruning bits from the sub-source 1 only. Let j_i' be the number of bits allocated to sub-source i under the new allocation. Since the quantizer function for the sub-source 1 is assumed convex, we prune off only one bit, that is $j_1' = j_1 - 1$. Then the new D' and R' , the new average distortion and rate, are

$$D' = p_1 d_i(j_1') + \sum_{i=2}^{M} p_i d_i(j_i) \tag{2.32}$$

and

$$R' = p_1 j_1' + \sum_{i=2}^{M} p_i j_i \ . \tag{2.33}$$

Now, the slope of the convex hull is

$$\frac{\Delta D_{\text{overall}}}{\Delta R_{\text{overall}}} = \frac{D' - D}{R' - R} \tag{2.34}$$

Now, the algorithm can be stated as follows.

1. For $i = 1, 2, \cdots, M$, set $B_i = q$. This is the initial bit allocation.

2. Calculate, for $i = 1, 2, \cdots, M$, for $j = 1, 2, \cdots, q$,

$$\begin{aligned} S_i(j, j-1) &= -\frac{\Delta D_{\text{overall}}}{\Delta R_{\text{overall}}} \\ &= -\frac{d_i(j) - d_i(j-1)}{j - (j-1)} \end{aligned} \tag{2.35}$$

3. Determine the sub-source for which $S_i(B_i, B_i - 1)$ is the lowest. Assume it is the sub-source l. Set $B_l = B_l - 1$.

4. Calculate the new overall average rate and distortion D and R as

$$D = \sum_{i=1}^{M} p_i d_i(B_i) \tag{2.36}$$

and

$$R = \sum_{i=1}^{M} p_i B_i \ . \tag{2.37}$$

Check if $R = 0$; if so, stop.

5. Repeat Steps 3) and 4).

A complete description of the algorithm and its analysis can be found in Riskin (1991) and Gersho and Gray (1992).

2.5.3 Entropy Coding

This section deals with the third block on the transmitter side of the Fig 2.4, viz., entropy coding the quantizer indices. Entropy coding refers to the class of techniques that aim to compress the signal without trading off any information in the process and hence fall under the category of lossless compression (cf Section 2.5.2). Consequently, it is used in applications where perfect fidelity needs to be maintained between the compressed and original signals. Such techniques are used in medical image compression, text compression etc. The use of such algorithms is not limited to the above mentioned applications; rather, they can be used to provide more efficient compression in schemes that use lossy compression as well. In this chapter, they will be used in conjunction with other lossy compression strategies.

Consider a discrete random variable X, with finite alphabet size of N, $\{x_0, x_1, \cdots, x_N\}$. The random variable X is characterized by its probability mass function,

$$p_X(x_i) = \Pr(X = x_i) \quad i = 0, 1, \cdots, N-1 \,. \tag{2.38}$$

The zeroth order entropy of the random variable X, $H(X)$, is defined as

$$H(X) = -\sum_{i=0}^{N-1} p_X(x_i) \log_2(p_X(x_i)) \,. \tag{2.39}$$

Entropy is measured in bits. Entropy coding gets its name from the fact that the source entropy provides a minimum bound for the average codeword length in the case of a memoryless source.

Entropy codes achieve compression by allocating bits to source symbols in accordance with their probability of occurrence and have been extensively studied in the literature (Blahut 1987). While traditionally, entropy codes have been variable length in nature, fixed length entropy codes have been popularized recently (Llados-Bernaus and Stevenson 1998) for error-resilient coding applications. A practical entropy code must be uniquely decodable so that there is only one possible sequence of codewords

for any input sequence. In some applications, it is also preferred to have instantaneous decodability, which implies that any codeword can be decoded upon receipt without having to wait for other codewords. The condition for instantaneous decodability, is that no codeword should be a prefix of another. For this reason instantaneous codes are also called prefix codes. In this section we will describe two popular entropy codes and discuss their usefulness in our JSCD codec.

Huffman Codes

An example of lossless compression algorithm is the Huffman code (Huffman 1951). This algorithm is based on the idea that it is more economical to assign fewer bits to the symbols that occur more often in the signal. The symbols are first sorted in the decreasing order of probability. The two symbols with the smallest probability are assigned codewords of equal length, differing in only the last bit. Now, these two symbols are grouped together to form one symbol with probability equal to the sum of the probabilities of the two individual codewords. The new source, with the reduced alphabet, is again sorted as before and the assignment process is repeated. This sorting and assigning process is continued until there are only two symbols remaining in the reduced source. One of the symbols is assigned a binary "1" and the other is assigned a binary "0". By working backwards and ungrouping the symbols, the complete code can be worked out. An example source probability distribution along with the grouping and the resulting Huffman code is shown in Figure 2.8. Note, that in some cases, the alphabets may be re-ordered so as to produce a code with different variance of the lengths in the Huffman code.

The Huffman algorithm is optimal in the sense that the average length of the Huffman code for any source is within one bit of its zeroth order entropy $H(S)$ (proof can be found in various textbooks like Sayood (1996) and Abramson (1963)). The average length can usually be further reduced by blocking a few symbols together. In fact, when n symbols are combined together, it can be shown that the average length, R, of the Huffman code is bounded, as in the following equation:

$$H(S) \leq R \leq H(S) + \frac{1}{n} \ . \tag{2.40}$$

Hence, we see that as n, or the number of symbols that are blocked together, increases, the average length of the code moves closer to the entropy of the source.

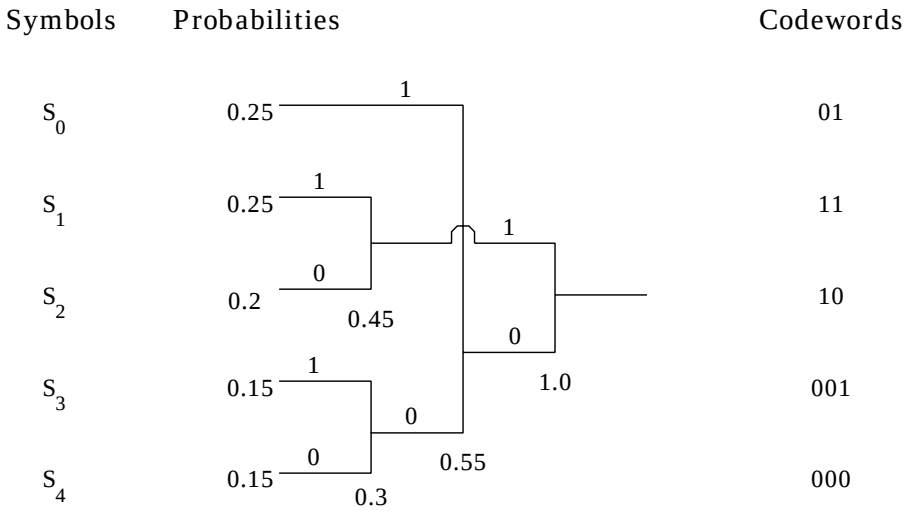


Figure 2.8: An Example Huffman Code

Huffman codes are used in image coding standards like JPEG and, since it can be implemented as a table look-up, it can be directly used in the JSCD. We will use Huffman codes in all the succeeding chapters to test the JSCDs developed in this dissertation.

Arithmetic Codes

On the flip side, blocking more and more codewords together causes the alphabet size to grow exponentially and the Huffman encoder becomes less and less practical. Also, when codes are designed for sources with varying probabilities, adapting the code to these probabilities becomes difficult. Under these circumstances, it is essential to develop a different set of codes that can deal with encoding sequence of symbols more effectively than a Huffman code does. One such code is the arithmetic code.

We noted above that encoding *sequences* of symbols can be more effective than encoding *individual* symbols. However, in order to Huffman encode a particular sequence of n symbols, codewords are needed for all possible sequences of length n symbols. This causes an exponential increase in the size of the alphabet. Arithmetic

codes offer a way of assigning codewords to particular sequences without having to generate codewords for all possible sequences. In this technique a unique tag is generated for the sequence that needs to be encoded, which is then given a binary code. The encoding is done incrementally and hence it is also easy to make the encoder adaptive, as long as the initial probability density function is known at both ends.

While the idea of arithmetic coding was introduced in Shannon's work (1948), Peter Elias developed the first recursive implementation of this idea. This work was not published, but was mentioned in a book by Abramson (1963). This idea was further developed in a book by Jelinek (1968). Modern arithmetic coding was then developed in Pasco (1976), Rissanen(1979) and Rissanen and Langdon (1981). Arithmetic codes are popular now and have been used in codecs like SPIHT (Said and Pearlman 1996) and EBCOT (Taubman). A detailed description of arithmetic coding can also be found in Sayood (1996). What follows is a brief description of the algorithm using an example.

An arithmetic code represents a sequence of symbols by a sub-interval in the interval $[0, 1)$. The size of the sub-interval represents the probability of the sequence and its location in the unit-interval uniquely identifies it. Let us consider a set of 3 symbols and their probabilities as shown in Table 2.1. In this case, the unit interval

Symbols	**Probabilities**	**Intervals**
S_0	0.7	$[0, 0.7)$
S_1	0.1	$[0.7, 0.8)$
S_2	0.1	$[0.8, 0.9)$
S_3	0.1	$[0.9, 1.0)$

Table 2.1: Symbol Probabilities and Corresponding Intervals

is split into four parts and the sub-intervals corresponding to each of the symbols is also tabulated in the Table 2.1. Let us assume that the sequence $\{S_0, S_1, S_2, S_0\}$ is to be encoded. Since the sequence starts with S_0, the sub-interval we begin with is $[0, 0.7)$. This interval is now split exactly according to the probabilities of the symbols and hence into $[0 \times 0.7, 0.7 \times 0.7)$, $[0.7 \times 0.7, 0.8 \times 0.7)$, $[0.8 \times 0.7, 0.9 \times 0.7)$ and $[0.9 \times 0.7, 1.0 \times 0.7)$; or in other words, into the sub-intervals $[0, 0.49)$, $[0.49, 0.56)$, $[0.56, 0.63)$ and $[0.63, 0.7)$. The first interval corresponds to the sequence $\{S_0, S_0\}$,

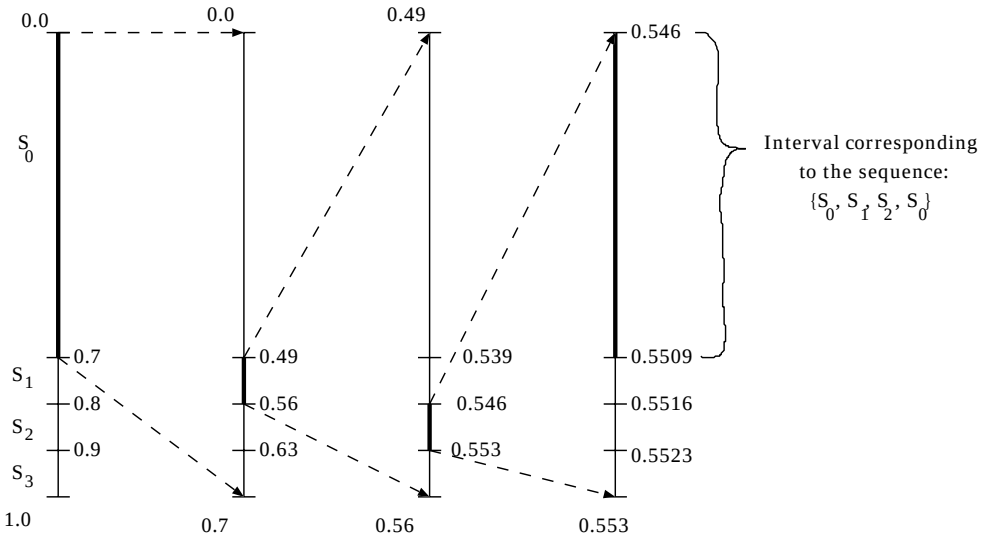


Figure 2.9: Splitting the Unit Interval for the Sequence: $\{S_0, S_1, S_2, S_0\}$.

the second to $\{S_0, S_1\}$, the third to $\{S_0, S_2\}$ and the fourth to $\{S_0, S_3\}$. Since the second symbol to be encoded is S_1, we now concentrate on the interval $[0.49, 0.56)$. By proceeding along similar lines we will see that the interval corresponding to the sequence we need to encode is $[0.546, 0.5509)$. The tag for this sequence will be 0.54845, if we assume that the mid-point of the interval is used as the tag. The progressive splitting of the unit interval for this particular example is depicted in Fig 2.9.

The interval corresponding to the entire string needs to be converted into a set of bits that can be transmitted digitally. We notice that the intervals are disjoint for different sequences. Hence, in order to identify the sequence, one may pick any point in this interval as a tag. Popular choices are the mid-point or the lower limit of the interval. The tag is then converted to its binary representation and transmitted. It is preferable to encode the sequence incrementally, and not have to wait until the last symbol in the sequence has been determined. In order to do this, the encoder waits until the cumulative probability of the sub-sequence is contained in either half of the

unit interval; viz., $[0, 0.5)$ or $[0.5, 1.0)$. The tag, at this point, is forever confined to either of these intervals. If the tag is confined to the upper half of the interval, then the tag number has to be greater than 0.5 and hence its first bit has to be 1; if the tag is less than 0.5, then the first bit is 0 (Sayood 1996). Once, the first bit has been transmitted, the encoder waits for the probability to again be confined to either half of the half-interval before transmitting the refinement bits. This procedure is repeated until all symbols are encoded. Often, since the system precision is finite, the intervals are scaled back to the unit interval after it reaches some small value. Scaling leads to loss of the most significant bit (MSB); however, this is not a problem, since the information has already been sent to the decoder.

We now describe the decoding process with the same example as before. When the decoder receives the tag value of 0.54845. The decoder successively chooses the interval that contains the tag completely in it. In effect, the interval splitting shown in Fig 2.9 is carried out at the decoder end as well. When the above tag value is received, the decoder picks the interval $[0, 0.7)$ as the first "approximation" as it is the interval in which the tag value is contained. Hence the first symbol has to have been S_0. Now this interval is split into smaller intervals corresponding to the four symbols as in Fig 2.9. On comparing the updated intervals to the tag value, we see that the interval $[0.49, 0.56)$, (corresponding to the symbol S_1) contains the tag value. Hence the second symbol in the sequence is S_1. The decoder goes on in this fashion until it completes the decoding process. Note that in this case, the decoder knows when to stop, because it knows how many symbols have been transmitted. Another way to provide a termination point to the decoder would be to have an end-of-transmission symbol.

We end this sub-section with a note on the error sensitivity and the applicability of the arithmetic codes to the JSCD designed in this dissertation. As seen from the description of the algorithm, arithmetic codes are implemented incrementally, in practice. This means that a single bit error can cause a catastrophic failure in the decoding of the rest of the string. If arithmetic codes are to be implemented as a table look up procedure, then the sequence must be split into smaller sequences for which codes can be generated and stored at the decoder end. However, for the MAP decoder proposed in this dissertation, maximum improvement can be expected only if there is sufficient memory between the symbols in the sequence and the source pdf

is non-uniform. Increasing the number of symbols to form a sub-sequence essentially makes the inter-symbol memory inaccessible to the MAP decoder. On the other hand, reducing the number of symbols to be combined does not allow the arithmetic coder to perform at its best. Hence, there is a conflict of requirement between the encoder and the decoder and for these reasons, it may not be beneficial to design a system which uses an arithmetic code to test the applicability of our JSCD decoder. Hence we use Huffman codes throughout this dissertation.

2.6 Test Sources

This final section briefly describes the test sources that will be used in this dissertation. The MAP decoders proposed in this dissertation are tested on sources with and without memory. The memoryless source used in this dissertation is a simple Gaussian source, where the source symbols are distributed according to a Gaussian pdf. Among the sources with memory we use natural images like those shown in Chapter 5 and auto-regressive (AR) sources.

A zero mean random sequence $X(n)$ is called an AR process of order p when it can be generated as the output of the system described by the equations (Jain 1989)

$$X(n) = \sum_{k=1}^{p} \rho(k) X(n-1) + \sigma(n) \quad , \forall n \tag{2.41}$$

$$E\{\sigma(n)\} = 0; \;\; E\{\sigma(n)X(m)\} = 0, \; m < n \tag{2.42}$$

where, $\sigma(n)$ is a stationary zero mean input sequence that is independent of the past outputs. This system uses the most recent p outputs and the current input to generate the next output recursively. Note that the noise process ($\sigma(n)$) and the AR process are uncorrelated. When $p = 1$, we get the first order AR process denoted by AR(1). When the noise process $\sigma(n)$ is Gaussian, the AR(1) process becomes a Gauss-Markov process. We will be using the Gauss-Markov process as a test source for the MAP decoders in the third and fourth chapters.

Chapter 3

MAP Decoding: Binary Symmetric Channels

As discussed in the previous chapter (Section 2.1), the problem of decoding a sequence of symbols transmitted over a noisy channel can be cast as a multiple decision problem. Using the maximum a posteriori probability (MAP) metric then becomes equivalent to minimizing the probability of decision error for the sequence. This fact inspired the development of a MAP decoder (Phamdo and Farvardin 1994) for fixed length encoded sources over noisy channels. The MAP decoder searches over all possible transmitted sequences to find the most probable one among them, which is declared as the best estimate of the transmitted sequence. Since the number of possible sequences is typically large, calculating the probability of occurrence of each sequence and finding the most probable one is time consuming. Hence, MAP problems are usually cast in a *dynamic programming* (DP) format, where some sequences that are obviously less probable are eliminated in the earlier stages of the algorithm, leading to a reduction in the complexity of the search.

For fixed-length encoded sources, the dynamic programming formulation is relatively simple. In this case, the MAP decoder algorithm uses a state space consisting of N states, where N is the number of codewords in the codebook. The algorithm then processes k bits at a time, where k is the dimension of the codewords. For variable-length encoded sequences, the partitioning of the bit stream into component codewords is not obvious and this makes the dynamic programming formulation more difficult. However, once an appropriate state-space is defined this problem can be

solved using a trellis based algorithm that prevents the complexity from escalating with time. In the independent work on variable-length sources done by Park and Miller (1998a) and Demir and Sayood (1998) the complexity of the exact MAP decoder increases with time, because a compact trellis structure is not used. In this chapter, a new state-space structure is proposed (Subbalakshmi and Vaisey 1998; Subbalakshmi and Vaisey 1999a), to deal with the variable-length codes in an elegant manner.

3.1 MAP for Entropy Coded Markov Sources over BSC

First we consider the problem of decoding an entropy coded, discrete Markov source transmitted over a BSC. Figure 3.1 depicts the coding scenario, in which a Markov source is quantized and the quantized indices are entropy coded and transmitted over a BSC. The MAP decoder for a BSC (MAP-BSC) uses the transition probabilities of the different source symbols and the channel cross over probability of the BSC to determine the best possible index sequence that was transmitted. Thus, the use of the source transition probabilities by the decoder is akin to using the residual redundancy present in the form of memory at the decoder end. In the following subsection, we state the MAP problem formally for a Markov source and also specialize this formulation to deal with memoryless sources.

3.1.1 The State Space and the MAP Problem

Let $\mathbf{C}$ denote the set of all possible B-bit sequences of variable-length codewords. The α^{th} such sequence is then denoted by $\mathbf{c}_\alpha^T = \{c_{\alpha,i}\}_{i=1}^T$, where $c_{\alpha,i}$ is the codeword corresponding to the i^{th} symbol in the transmitted stream and T is the total number of codewords in the stream. Since there are many ways in which to partition the received (variable-length encoded) bit stream to yield different index sequences, the problem is to find the "best" possible index sequence, given the source and channel statistics. We note that the different partitions of the received bit stream will, in general, lead to different number of codewords in the index sequence. Let $\mathbf{R}$ denote the set of all B-bit received streams and let the sequence $\mathbf{r}_j^{n(j)}$ represent the j^{th} stream, with

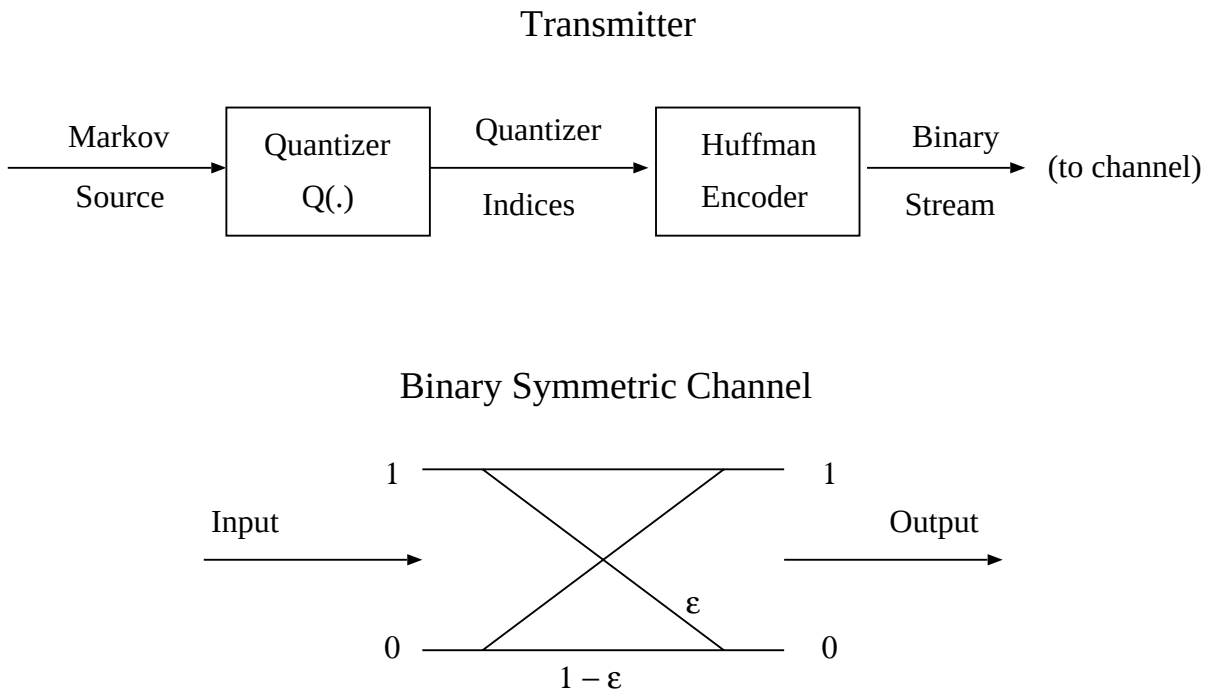


Figure 3.1: Coding Scenario

$n(j)$ being the number of codewords in the stream. We also define the probability of the symbol s_i to be $\Pr(s_i)$ with $\Pr(s_i|s_k)$ representing the probability that s_i was transmitted immediately after s_k. If $\hat{j}$ represents the index of the most probable transmitted sequence, then our problem is to determine

$$\mathbf{c}_{\hat{j}}^{n(\hat{j})} = \arg \max_{\mathbf{c}_j^{n(j)}} \Big\{ \Pr(c_{j,1}) \epsilon^{H_d[c_{j,1}, r_{j,1}]} (1-\epsilon)^{(l_1 - H_d[c_{j,1}, r_{j,1}])} \times \prod_{k=2}^{n(j)} [\Pr(c_{j,k}|c_{j,(k-1)}) \epsilon^{H_d[c_{j,k}, r_{j,k}]} (1-\epsilon)^{(l_k - H_d[c_{j,k}, r_{j,k}])}] \Big\} \tag{3.1}$$

where $H_d[c_{j,k}, r_{j,k}]$ is the Hamming distance between the k^{th} transmitted codeword of a specific sequence $\mathbf{c}_j^{n(j)}$ and the k^{th} word of the received sequence $\mathbf{r}_j^{n(j)}$ and l_k is the length of these words.

We propose a novel state space for the MAP-BSC decoder consisting of two classes of states: the *complete* and the *incomplete* states. We describe these states using an example variable-length codebook: $\{A:0, B:10, C:110, D:111\}$. Let the first three bits of the transmitted bit stream be $\{0,1,0\}$ corresponding to the sequence A, B and the received bit stream be $\{0,1,1\}$. We see that the bit stream can be partitioned in seven ways, as shown in Figure 3.2. The dots connected by solid lines represent a single completely received codeword, those connected by dotted lines denote bits of a codeword that are only partially received and the dark dots represent the one bit codeword. The first four cases (cases 1, 2, 3a and 3b) represent the situations when the received bits are completely partitioned into full codewords and hence these cases correspond to *complete states*. Cases 4, 5 and 6 correspond to situations in which the received bits cannot be completely partitioned into full codewords, and hence they represent the *incomplete states*. The *degree of incompleteness* is defined as the number of bits of the incomplete codeword that have been received so far. The state space thus consists of six states: 4 degree 0 states (one for each codeword), a degree 1 and a degree 2 state.

We note that the number of degrees of incompleteness and the number of transitions between the states are determined by the lengths of the codewords in the codebook. For a codebook containing N codewords of varying lengths belonging to the set, $\mathcal{L} = \{l_{\min}, \cdots, l_{\max}\}$, the maximum degree of incompleteness is $l_{\max} - 1$. State transitions from degree $k-1$ states to degree k states ($k > 0$) are possible at every

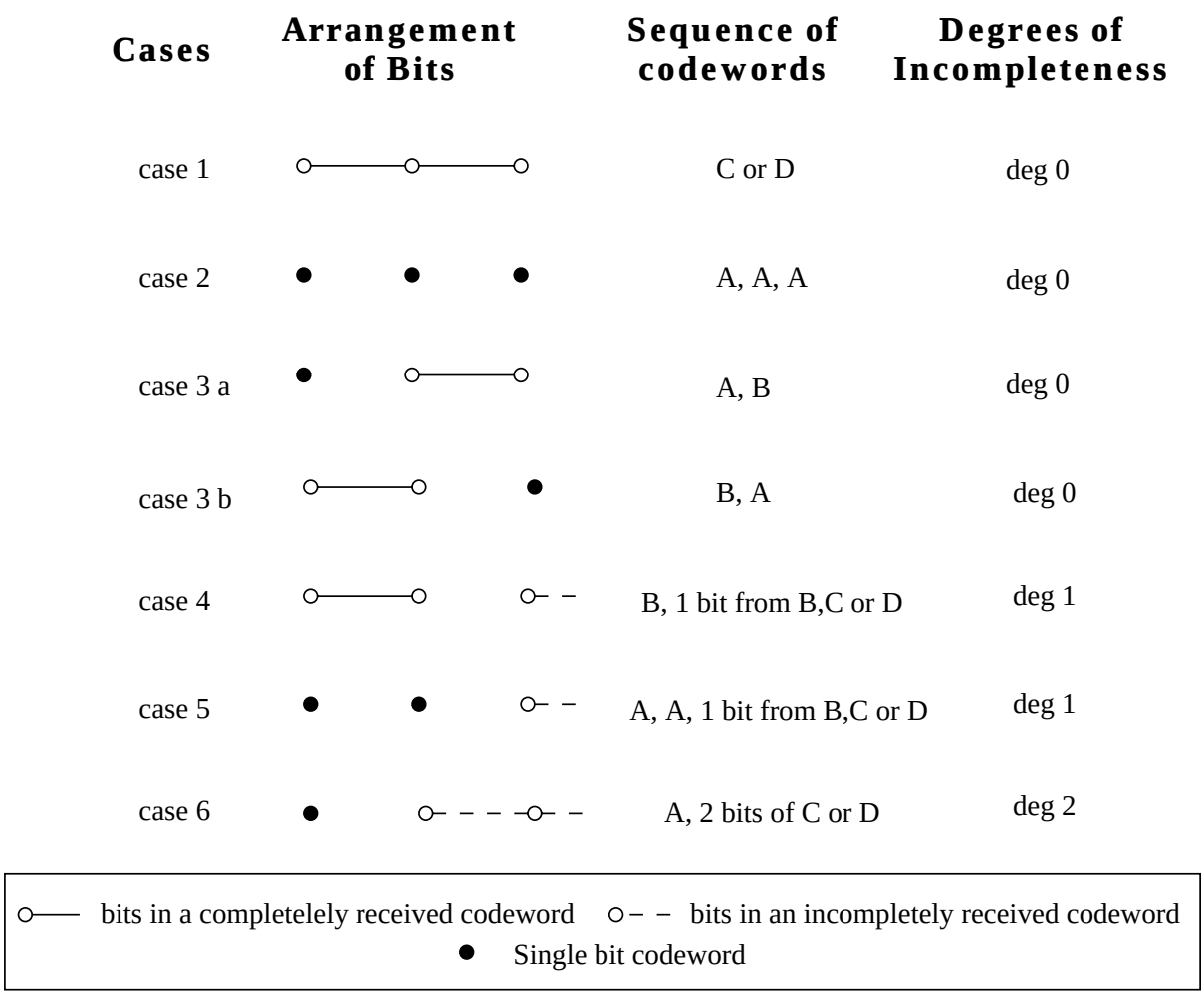


Figure 3.2: Degrees of Incompleteness in the State Space

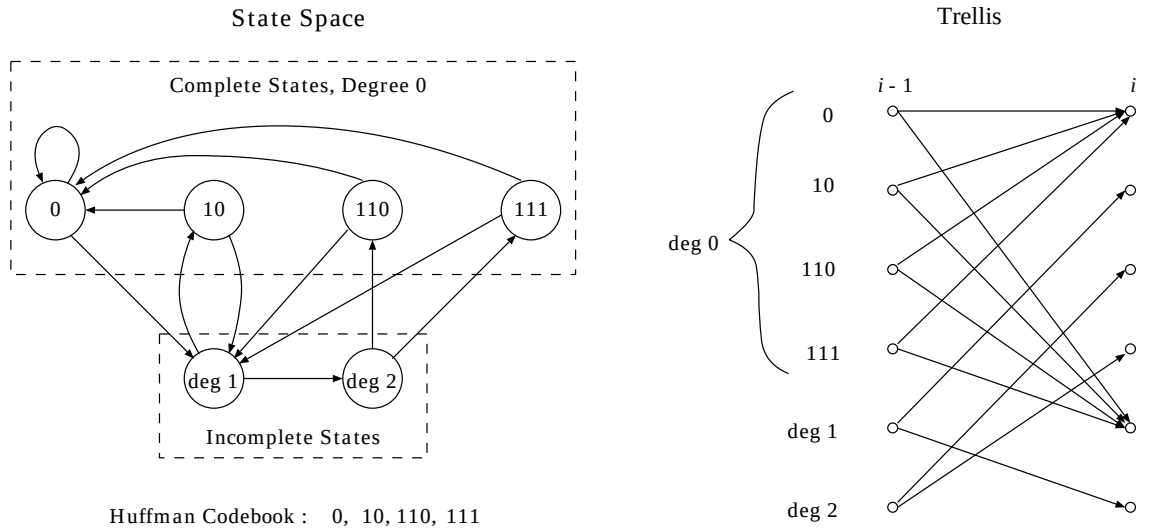


Figure 3.3: MAP-BSC: State-Space and Trellis Diagram for the Codebook $0, 10, 110, 111$.

stage of the algorithm; however, a transition from a degree k ($k > 0$) state to one of the degree 0 states is possible only if $k = l_i - 1$, where l_i is the length of the codeword represented by that degree 0 state. A trellis is used to show the evolution of the state space with time (as bits are received). At each stage of the trellis there are as many nodes as there are states; the state space and the transitions from state $i-1$ to i are depicted in Figure 3.3.

3.1.2 The MAP-BSC Algorithm for a Markov Source

The key feature of the algorithm, described below, is the use of complete and incomplete states. Two operations take place at each stage of the decoding algorithm. One consists of examining the metrics of all the paths, entering each node (state-stage pair) in the trellis and the other consists of looking for a merger of the paths and the actual declaration of decoded codewords (if there is a merge). For complete states, the path update step involves finding the best metric path to the state and retaining

it. For the incomplete states, we must retain the metric values of all the paths back to the last complete state. Note that since the logarithm is a monotonically increasing function, maximizing the logarithm of the a posteriori probability will be the same as maximizing the a posteriori probability. Hence, we will maximize the logarithm of the a posteriori probability of the sequence, a quantity which will be referred to as the "metric" from now on.

Let $\mathbf{v} = (v_x, v_y)$ represent the node corresponding to state v_x at stage v_y. The complete states are numbered from 1 to N corresponding to the N codewords in the codebook and incomplete states are numbered from $N+1$ to $N + l_{\max} - 1$. Let c_k denote the length d_k codeword corresponding to the complete state k and let $\mathbf{b}_{k,i}$ be the d_k most recent received bits at node (k,i). $\mathcal{M}[k,i,c_j]$ gives the metric increment associated with the path segment that begins at node $(j,(i-d_k))$ and terminates in the node (k,i) (where k and j are complete states). Thus, when $i > d_k$

$$\mathcal{M}[k,i,c_j] = \log_{10}(\Pr\{c_k|c_j\}) + H_d[c_k, \mathbf{b}_{k,i}]\log_{10}(\epsilon) + (d_k - H_d[c_k, \mathbf{b}_{k,i}])\log_{10}(1-\epsilon)$$

Now, $M_{k,i}$ is defined to be the metric value associated with the maximum metric path terminating in (k,i), where $k \in \{1, 2, \cdots, N\}$. For the incomplete states we define $M_{k,i}(u)$ to be the u^{th} element in the vector of metric values that need to be remembered at (k,i), where $k \in \{N+1, N+2, \cdots, N+l_{\max}-1\}$ and $u \in \{1, 2, \cdots, N\}$. Finally, let $\mathbf{v}^p$ be the parent of the node $\mathbf{v}$, where a parent is defined as the penultimate node in the maximum metric path terminating at node $\mathbf{v}$ and let $d_{v_x^p}$ be the length of the codeword corresponding to the parent node (which is complete). The algorithm can now be stated as follows:

Initialize:

Input first $l_{\min} - 1$ bits

For $i = 1, \cdots, l_{\min} - 1$,

$M_{k,i} \leftarrow 0 \ \forall k \in \{1, 2, \cdots, N\}$.

$M_{k,i}(u) \leftarrow 0 \ \forall \ u \in \{1, 2, \cdots, N\}; \forall k \in \{N+1, \cdots, N + l_{\max} - 1\}$.

$i \leftarrow l_{\min}$

For $k = 1, 2, \cdots, N$

$\mathbf{v} = (k, i)$

$\mathbf{v}^p = \Phi$ (no parents)

Input:

Input bit at the i^{th} stage

Update path metrics:

For $k = 1, 2, \cdots, N$

$$M_{k,i} \leftarrow \begin{cases} \max_j\{M_{(N+d_k-1),(i-1)}(j) \ + \ \mathcal{M}[k, i, c_j]\}, \ i > d_k \\ \log_{10}(\Pr\{c_k\}) + H_d[c_k, \mathbf{b}_{k,i}] \log_{10}(\epsilon) + (d_k - H_d[c_k, \mathbf{b}_{k,i}]) \log_{10}(1-\epsilon), \ i = d_k \\ 0 \ , \quad i < d_k \end{cases}$$

$$M_{k,i}(u) \leftarrow \begin{cases} M_{u,(i-1)} & \forall \ u \ \in \{1, 2, \cdots, N\}, \ k = N+1. \\ M_{(k-1),(i-1)}(u) & \forall \ u \ \in \{1, 2, \cdots, N\}, \ \forall k \in \{N+2, \cdots, N + l_{\max} - 1\} \end{cases}$$

$$j_k^* \ \leftarrow \ \arg \ \max_j\{M_{(N+d_k-1),(i-1)}(j) + \mathcal{M}[k, i, c_j]\}, \ \forall k \ \in \{1, 2, \cdots, N\}.$$

We note that updating $M_{k,i}(u)$ involves only a copy operation whereas updating $M_{k,i}$ involves some calculations.

Update paths:

For $k \in \{1, 2, \cdots, N\}$

$\mathbf{v} = (k, i)$

$v_x^p = j_k^*; \ \ v_y^p = i - d_{j_k^*};$

Merge Check and Output:

A merge is declared when all nodes at stage i arise from a common ancestor. For degree zero nodes, we trace back from the respective nodes at stage i; however, for any incomplete state k, we trace back from *all* of the complete states of stage $i-k+N$. This is so because at incomplete states no parent node can be discarded as it may be the parent of some state at a later stage. In order to determine if there are merges we define a set of nodes $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_{Nl_{\max}}\}$ and initialize them to the nodes from which we need to trace back in order to determine a merge. We need $Nl_{\max}$ such nodes since each degree of incompleteness contributes to N nodes and the N complete nodes contribute to one parent each.

$\mathbf{V}$ is formed according to

$\mathbf{v}_m = (m - N\lfloor \frac{m}{N} \rfloor,\ i - \lfloor \frac{m}{N} \rfloor),\ \forall m \in \{1, 2, \cdots, Nl_{\max}\}.$

If $(i \geq l_{\max} + 1)$

a: *If* (the parents of all the nodes in $\mathbf{V}$ are the same)

decide that the path terminating in $\mathbf{v}_1$ was indeed transmitted

and set the parent of all the nodes at this stage to be Φ.

Else

$\mathbf{v}_m \leftarrow \mathbf{v}_m^p,\ \forall\, m\ \in \{1, 2, \cdots, Nl_{\max}\}$

If $(\mathbf{v}_m^p \neq \Phi\ \ \forall\, m\ \{1, 2, \cdots, Nl_{\max}\})$

go to **a**.

If (end of sequence)

$k^* \leftarrow \arg\ \max_{k \in \{1,2,\cdots,N\}} M_{k,i}$

Declare the path terminating at (k^*, i) as the optimal path.

Else

$i \leftarrow i + 1$

go to **Input**.

The complexity of any trellis based algorithm is generally considered to be equal to the number of states in the state-space. In this case, the complexity would be $N + l_{\max} - 1$. Note that as N increases, the number of complete states increases linearly, but the increase in the number of incomplete states depends on the length of the longest codeword. Also, the operations in each of the states differ in complexity: at the complete states, we perform a compute and compare operation on the partial metrics, although, at the incomplete states, we only retain path metrics which corresponds to a copy operation which is not as complex as the operations at the complete states. Finally, there is the back-tracking procedure which also contributes to the complexity. As we noted in the description of the algorithm, back-tracking takes place from both the complete and incomplete states. Back-tracking from the incomplete states effectively translates to back-tracking from the parents of these states, which implies back-tracking from all the complete states of previous stages. So, in effect, we need to back-track from $N \times l_{\max}$ complete states. For example, if N were doubled, and we make an assumption that the length of the longest codeword is also doubled, then the complexity due to the back-tracking operation also doubles. The

total complexity thus scales approximately linearly with N.

It is also possible to decrease the complexity of the algorithm further, when the greatest common divisor, g, of the lengths of the codewords in the codebook is greater than one. In this case, a codeword can never be complete unless the number of bits is a multiple of g; so, we may consider g samples at a time. Now, the degree 1 state corresponds to the condition where the last g bits are not yet part of a complete codeword; the degree 2 state corresponds to the situation when the last $2g$ bits are not yet part of a complete codeword and so on. This causes a reduction in the number of states, which will now be $N + \frac{l_{\max}-1}{g}$.

3.2 MAP-BSC for Memoryless Sources

In the case of sources without memory, the MAP metric in Eqn 3.1 reduces to

$$\mathbf{c}_{\hat{j}}^{n(\hat{j})} = \arg \max_{\mathbf{c}_j^{n(j)}} \prod_{k=1}^{n(j)} [\Pr(c_{j,k}) \epsilon^{H_d[c_{j,k}, r_{j,k}]} (1-\epsilon)^{(l_k - H_d[c_{j,k}, r_{j,k}])}] \tag{3.2}$$

where the symbols have the same meaning as before. We note that the conditional source probability term is replaced by the individual symbol probabilities, because the source is memoryless. Hence the order of the source symbols in the sequence does not matter any more and it is not necessary to remember the predecessor to each source symbol in the sequence. The result is that a single degree zero (complete) state may be used to represent all codewords. Figure 3.4 illustrates the state space and trellis diagram associated with the MAP decoder for memoryless sources, using the same example codebook as before.

The MAP decoder algorithm for memoryless sources differs from that for Markov sources in that it uses a different metric and the merge check step need only trace back from each of the states at the last stage. Before we go on to describe the algorithm formally, we need to define some slightly different notation. This step is necessary because we do not have a 1:1 mapping between codewords and complete states anymore.

- $c_n(d)$ denotes the n^{th} Huffman codeword of dimension d (as opposed to $c(s_k)$ which is the codeword corresponding to s_k). Note, that unlike in the case of

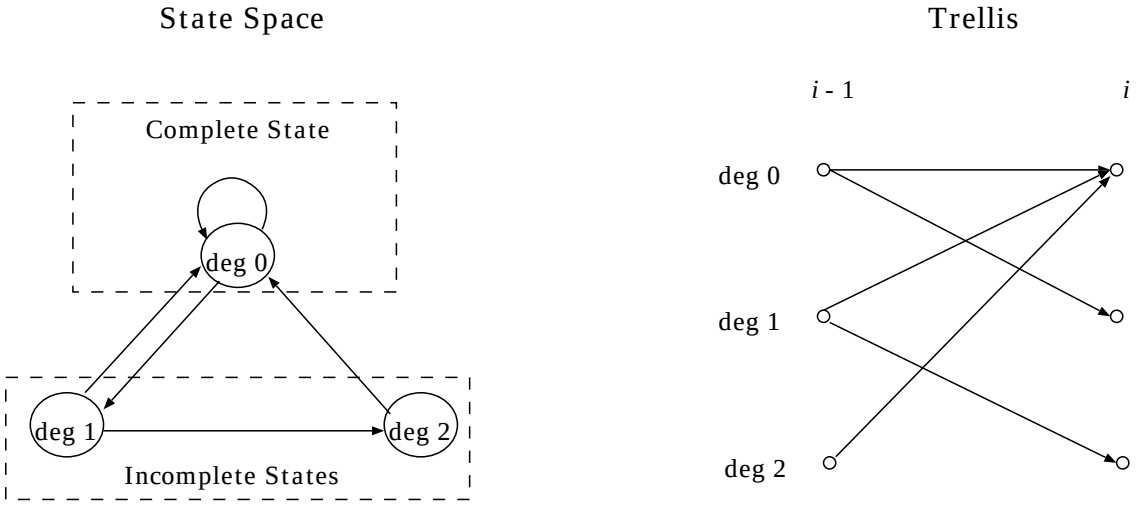


Figure 3.4: MAP-BSC State-Space and Trellis Diagram for the Codebook : $0, 10, 110, 111$ for a memoryless source.

sources with memory, the codewords are indexed according to their dimension and they are all represented by the same state (degree 0).

- $\mathbf{b}_{id}$ denotes the d most recent received bits at stage i.
- $\mathcal{M}[\mathbf{b}_{id}, c_n(d)]$ gives the metric increment associated with codeword $c_n(d)$ when the received bits were $\mathbf{b}_{id}$, or

$$\begin{aligned}\mathcal{M}[\mathbf{b}_{id}, c_n(d)] &= \log_{10}(\Pr\{c_n(d)\}) + H_d[c_n(d), \mathbf{b}_{id}] \log_{10}(\epsilon) \\ &\quad + (d - H_d[c_n(d), \mathbf{b}_{id}]) \log_{10}(1-\epsilon)\end{aligned}$$

- $E_{id}(k)$ is the k^{th} codeword in the sequence of codewords that forms the maximum-metric path of degree d at stage i.
- M_{id} is the metric value associated with the path of degree d at stage i. Note that unlike in the algorithm for Markov sources, we do not have a vector of path metrics to maintain for the higher degrees of incompleteness.

- W_d is the set of all codewords of dimension d and m^*_{id} is given by:

$$m^*_{id} = \max_{n \in W_d} \mathcal{M}[\mathbf{b}_{id}, c_n(d)].$$

- $\star$ denotes concatenation.

We can now state the algorithm as follows:

Initialize:

$i \leftarrow 1$

$M_{0d} \leftarrow 0 \ \forall \ d \in \{1, 2, \cdots, l_{max} - 1\}$

$E_{0d}(.) \leftarrow \{\phi\}$

Input:

Input bit at the i^{th} stage

Update path metrics:

$$M_{id} \leftarrow \begin{cases} M_{(i-1),(d-1)} & d \neq 0 \\ \max_j \{M_{(i-1),j} + m^*_{i,j+1}\} & d = 0 \end{cases}$$

$d^* \leftarrow \arg \max_j \{M_{(i-1),j} + m^*_{i,j+1}\}$

$n^* \leftarrow \arg \max_n \ \mathcal{M}[\mathbf{b}_{i,d^*+1}, c_n(d^* + 1)]$

Update paths:

$$E_{id}(.) \leftarrow \begin{cases} E_{(i-1),(d-1)}(.) & d \neq 0 \\ E_{(i-1),d^*}(.) \star n^* & d = 0 \end{cases}$$

Merge check and Output:

If $i \geq l_{max} + 1$

a: If $E_{i0}(0) = E_{i1}(0) \cdots = E_{il_{max}}(0)$

decide that the codeword $E_{id}(0)$ was indeed transmitted

update $E_{id}(k) \leftarrow E_{id}(k+1) \ \forall \ k, \ d$

go to **a**

If end of sequence

declare $E_{id}(.)$

Else

$i \leftarrow i + 1$

go to Input

Since the state-space is now reduced, with the number of states equal to $l_{\max}$, the complexity of the algorithm is less than that of the MAP decoder for Markov sources. The MAP decoder for memoryless sources can, thus, be considered as a special case of the MAP decoder for sources with memory.

3.3 Do the Algorithms Find the Exact MAP Path?

It can be shown by recursion that the algorithms developed in this chapter determine the exact MAP sequence. We consider the MAP-BSC algorithm for a Markov source only, since similar arguments can be presented for the MAP decoder for memoryless sources. In each stage, we will show that the algorithm does not discard any path segment that is part of the optimal path and that every new segment added to each of the candidate paths is part of the optimal path terminating in that node.

At the first stage, the algorithm takes the first $l_{\min}$ number of samples, determines the MAP metric associated with each $l_{\min}$ dimension codeword and stores it as the best path terminating in that codeword at this stage. Since there is only one candidate path and this is chosen, it is optimal. Since in this step we do not eliminate any path, the optimal path has not been discarded. Now, we assume that the algorithm proceeds to do the exact MAP estimate until the i^{th} stage and prove that at the $(i+1)^{\text{th}}$ stage it still retains the exact MAP path.

At the $(i+1)^{\text{th}}$ stage, the algorithm performs two different operations: the path update and the merge check operations. The path update step for complete states involves finding the maximum metric path leading to *each* of the codewords from preceding stages. Since the algorithm has not eliminated the optimal MAP path until the i^{th} stage and has retained the best metric path to each codeword in the previous stages, finding the best segment leading to the codewords in the $(i+1)^{\text{th}}$ stage is still optimal. At each of the incomplete states, we simply retain the metrics of all the complete paths till then. This step is essential so as to allow for the first order Markov dependence in the source. Hence the path-update step at each stage is optimal.

Finally, a merge check is done from *both* the complete and the incomplete states at the $(i+1)^{\text{th}}$ stage, implying that we look for common ancestor paths from which the paths terminating in each state of the $(i+1)^{\text{th}}$ stage originate. This is then declared as part of the optimal sequence. This follows from the fact that all paths entering

each node at the $(i+1)^{\text{th}}$ stage are the optimal paths terminating in these nodes. Hence, the overall optimal path must be one of these paths. So the common ancestor segment is part of the overall optimal path.

3.4 Generalization to Markov Sources of Higher Orders

In this section, we briefly look at how this algorithm can be extended to Markov sources of higher orders. In general, trellis based algorithms are extended to higher orders of memory by designing a new state-space that consists of S^n states, where S is the number of states in the original algorithm and n is the order of the Markov source. In our case, we will show by example that while the number of complete states increases exponentially, the number of incomplete states does not.

Consider an example of a second-order Markov source represented by two codewords $A : 0$ and $B : 10$. As before, the MAP sequence is that which maximizes the probability of the transmitted sequence, given the received sequence. For a first-order Markov source, the MAP sequence was given by Eqn 3.1, since the conditional probability of any symbol depends only on the immediately preceding symbol. In a second-order Markov source, the conditional probability will depend on two preceding symbols. Hence we form a new set of "meta-symbols" each of which is a combination of 2 of the original symbols. In our example, we have four meta-symbols, corresponding to AA, AB, BA, BB. These meta-symbols have lengths $2, 3, 3$ and 4 respectively. Therefore the degrees of incompleteness that we need to define for this case will range from 1 to 3. Hence the new MAP decoding algorithm will have 4 complete states and 3 incomplete states. Note that while the number of complete states increased exponentially (from 2 to 4), the number of incomplete states increased by only 1.

The MAP problem for a second-order Markov source, transmitted over a BSC can be stated mathematically as

$$\mathbf{c}_{\hat{j}}^{n(\hat{j})} = \arg \max_{\mathbf{c}_j^{n(j)}} \Big\{ \Pr(c_{j,1}, c_{j,2}) \epsilon^{(H_d[c_{j,1}, r_{j,1}] + H_d[c_{j,2}, r_{j,2}])} (1-\epsilon)^{(l_1 + l_2 - H_d[c_{j,1}, r_{j,1}] - H_d[c_{j,2}, r_{j,2}])} \times \prod_{k=3}^{n(j)} [\Pr((c_{j,k}, c_{j,(k-1)}) | (c_{j,(k-2)}, c_{j,(k-3)})) \epsilon^{(H_d[c_{j,k}, r_{j,k}] + H_d[c_{j,(k-1)}, r_{j,k}])}$$

$$\times(1-\epsilon)^{(l_k+l_{k-1}-H_d[c_{j,k},r_{j,k}]-H_d[c_{j,(k-1)},r_{j,(k-1)}])}\Big]\Big\} \quad (3.3)$$

where the symbols have the same meaning as before. In general, for N codewords with maximum length $l_{\max}$, the number of states in the algorithm will be $N^2+2\times(l_{\max}-1)$. By extension, for an n^{th} order Markov source each meta-symbol is made up of n original symbols and the number of states in the state-space is $N^n + n \times (l_{\max}-1)$.

3.5 Experimental Results

In this section, we will discuss the various experiments that were performed and their results in order to evaluate the performance of the proposed decoders. The performance of the MAP-BSC is compared to that of the Huffman decoder (HD) using two parameters: the percentage of bits that are out of synchronization (PBOS) and the modified signal to noise ratio (MSNR) of the decoded stream, as described in Chapter 2 of this dissertation.

3.5.1 MAP-BSC for Markov Sources

To evaluate the performance of the proposed decoder, experiments were performed on 5000 samples of a zero-mean, 1st-order Gauss-Markov source (Section 2.6) using a range of correlation coefficients values, ρ_s; the underlying Gaussian source had unit variance. Note that at 5000 samples, the mean and variance of the source statistics have stabilized to within 1% of the theoretical values and the variation in the results for the different channel realizations (mentioned in greater detail, later in the section) is quite small. Also, although the correlations used in the experiments are often high, and could be reduced via prediction or linear transforms, the intent is to compare MAP-BSC to Huffman decoding in a situation where the amount of memory is significant but easily controlled. There are many applications where there may be significant memory, but little correlation; for example, the runs of zeros found in image subbands (Section 5.2.6).

To obtain a discrete system, the test sources were quantized using a mid-tread uniform quantizer, with the output-rate selected to provide a "reasonable" SNR under noiseless conditions. The symbols were then entropy coded using a table of Huffman codewords that were designed from large training sets having the same correlation

Symbol Probability	Codeword
0.21504	01
0.18616	00
0.18534	111
0.12072	101
0.11974	100
0.05823	1100
0.05815	11010
0.02853	110111
0.02810	110110

Table 3.1: Huffman Codebook and Source Probabilities for the AR(1) source of $\rho_s = 0.9$ quantized to 9 levels.

coefficients (ρ_s) as the sources. The codeword-lengths in the table ranged in size from 2 to 6 bits for the $\rho_s = 0.9$ and $\rho_s = 0.8$ sources at an average rate of 3 bits per sample. The Huffman table for the $\rho_s = 0.9$ source quantized to 9 levels is given in Table 3.1 along with the probabilities associated with the quantized source. We chose a simple quantizer structure, since the point of the experiments was only to gauge the performance of the MAP-BSC decoder and not to construct a state-of-the-art codec.

The codec was implemented using four separate programs. The first was the training stage, where an AR(1) source was generated using a user specified seed for the random number generator. This program also gathers the source statistics, designs the Huffman codebook and write these out to separate files. The second program essentially encodes the source. Five thousand samples of the same source (different seed for the random number generator) were generated and these were quantized by the same quantizer as before. The quantized source was Huffman encoded using the Huffman codebook written out by the previous program. The variable-length encoded bit stream is then written out to another file. This file is fed to the channel program which corrupts the transmitted bit stream. The channel program also allows the user to input the desired value for the channel bit error-rate and the random number seed that controls the channel realization. The output of this program is the corrupted transmitted stream, which will be referred to as the corrupted stream. The MAP decoder program reads in this corrupted stream, the value of the channel

cross-over probability, the files corresponding to the Huffman codes and the source statistics, and outputs the MAP decoded sequence. A Huffman decoder program was also separately implemented that reads in the corrupted stream and the Huffman codebook and outputs the Huffman decoded stream. The performance of the decoders was determined using an evaluation program that calculates the MSNR and PBOS values.

Figure 3.5 compares the decoded MSNR for both the decoders for the AR(1) source with $\rho_s = 0.9$, quantized to $N = 9$ with a step size of 1.25. Error-bars are used to show the variation between the different channel realizations. Results are presented over a range of channel error rates and the performance of the MAP-BSC is consistently superior to the HD. It is seen from the graph that this improvement is highest in the "middle" of the range of error rates, reaching a maximum of 8.32 dB at an error rate of about $10^{-1.5}$ and decreases towards either extremes of error rates. We also note that at very high error rates ($10^{-0.5}$ to $10^{-0.7}$) the performance of the MAP-BSC falls below that of the HD. It can be shown that when $\epsilon \to 0$, the MAP decoder tends to pick up the sequence that has the highest probability of occuring regardless of what it observes. This would imply that if this sequence was not actually transmitted, then the MAP decoder makes a mistake. It is difficult to arrive at the cost of this mistake in terms of the PBOS and MSNR analytically. Note, that the MAP decoder still does find the MAP sequence: the a posteriori probability of the sequence returned by the Huffman decoder is smaller than that of the sequence returned by the MAP decoder and hence the decoder is still optimal, only not in the sense of maximizing either of the metrics that are used as performance measures.

Figure 3.6 shows the average percentage of bits that are out of synchronization (PBOS) in the decoded sequence, for both the HD and MAP-BSC. The trends in the relative performance of the MAP-BSC versus the HD is just as in the MSNR profile, with the maximum improvement in percentage loss of synchronization being 20.94%. The implications of the decrease in PBOS varies with the application. In the case of image transmission, it would mean that the number of synchronization markers that need to be inserted in the signal can be reduced for any desired quality and for audio signals, this would imply fewer pops and clicks. It would also imply that the necessity for error-correcting codes can be substantially reduced.

Figures 3.7 and 3.8 depict the performance of the MAP-BSC decoder versus the

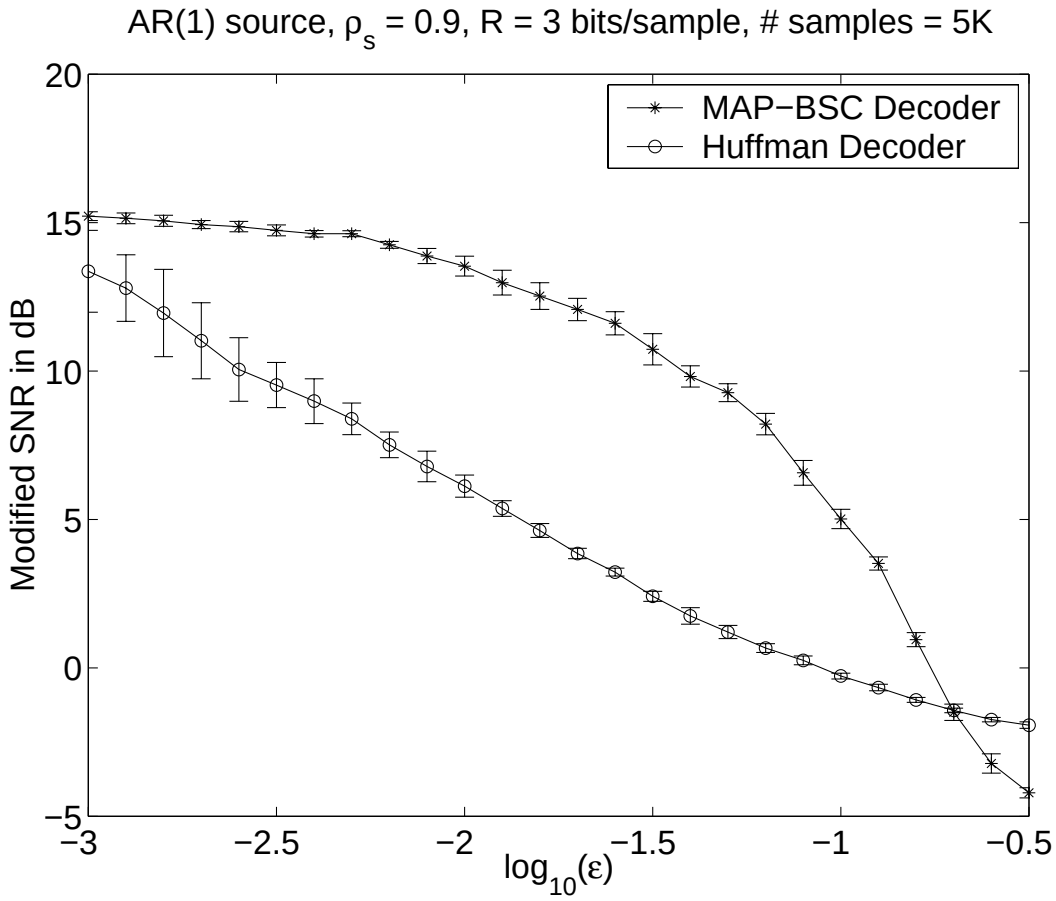


Figure 3.5: Comparison of the modified SNR of the MAP-BSC with the HD for an AR(1) source with $\rho_s = 0.9$, quantized to $N = 9$.

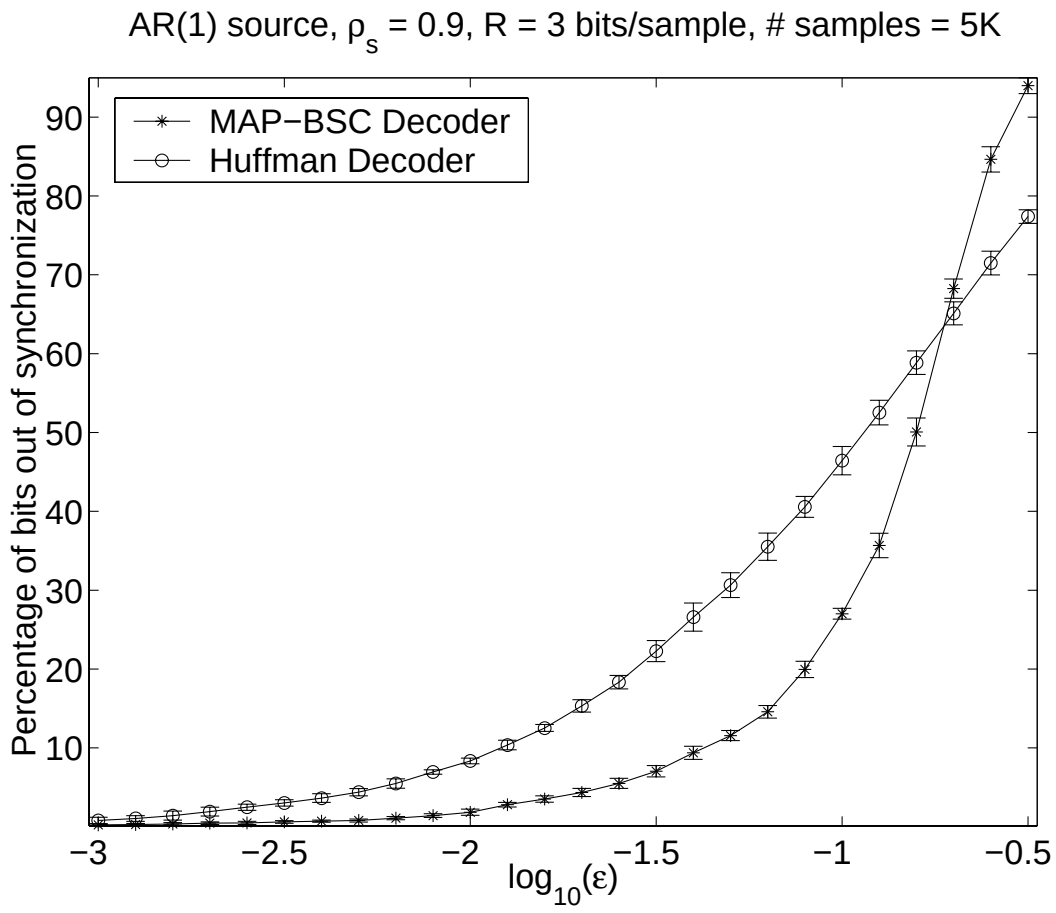


Figure 3.6: Comparison of the percentage loss of synchronization of MAP decoded sequence with that of the Huffman decoded sequence for an AR(1) source with $\rho_s = 0.9$, quantized to $N = 9$.

Huffman decoder in terms of the modified SNR and the PBOS respectively for 5000 samples of an AR(1) source of correlation coefficient 0.8, quantized with a $N = 9$ level quantizer at a rate of $R = 3$ bits per sample (step size 0.9). We note that just as in the source with $\rho_s = 0.9$, the MAP-BSC performs better than the Huffman decoder for most error rates. The trend in performance is the same, but the maximum improvement is smaller, at 4.73 dB in MSNR and 12.05% in the PBOS. This decrease in improvement is expected, since the residual redundancy available to the decoder is smaller for a source with smaller correlation coefficient. Experiments were also conducted with the correlation coefficient further reduced to $\rho_s = 0.5$. The performance trend is again the same, with the maximum improvement in MSNR and PBOS now being 1.17 dB (at $10^{-1.7}$) and 7.27% (at $10^{-1.3}$) respectively.

Finally, in order to focus on the way in which the performance varies with the source correlation, we fixed the error rate to $\epsilon = 10^{-1.5}$ and swept ρ_s. The MSNR and PBOS results of this experiment are shown in 3.9 and 3.10 and, as expected, the MAP-BSC improvement decreases with ρ_s, approaching zero for $\rho_s < 0.3$. This decrease in improvement is expected, since the residual redundancy available to the decoder drops with ρ_s.

Performance Under Mismatch Conditions

In practice, the estimation of channel error rates may be inaccurate. Hence, the performance of the proposed decoder was studied under statistical mismatch conditions with a view to gauging the robustness of the decoder towards inaccuracies in channel error rate estimation. The parameter $\Delta = \log_{10}(\epsilon_a) - \log_{10}(\epsilon_o)$, in the figures, refers to the difference between the logarithms of the assumed and true channel errors. In other words, it is a measure of how far the estimated value of the channel error rate (ϵ_a) is from the actual value (ϵ_o) in *orders of magnitude*. ϵ_a is also the error rate at which the MAP-BSC is designed to operate. A positive Δ indicates an over-estimation of errors and a negative Δ indicates an under-estimation of errors. So, the curve obtained when $\epsilon_o = \epsilon_a$ ($\Delta = 0$) represents the case when there is no error in the estimation of the channel error rates.

Figures 3.11 and 3.12 show the performance of the MAP decoder under various

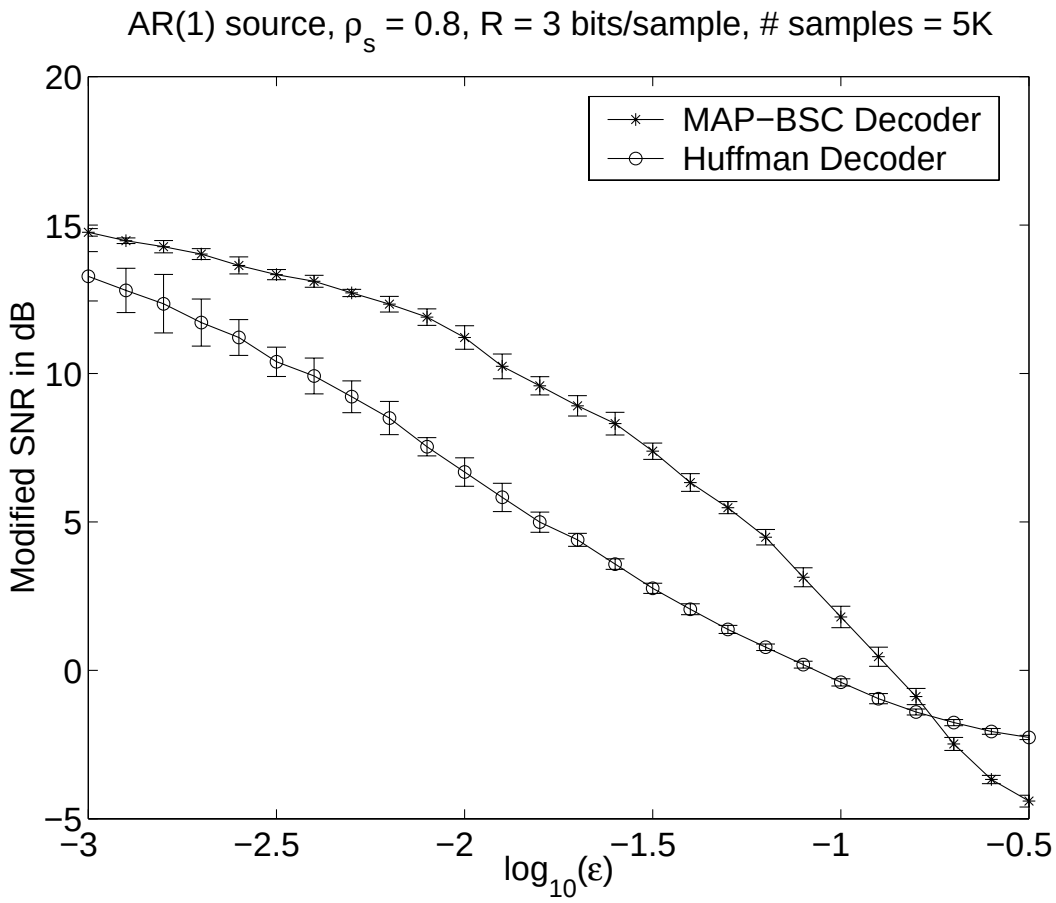


Figure 3.7: Comparison of the modified SNR of the MAP-BSC with the HD for an AR(1) source with $\rho_s = 0.8$, quantized to $N = 9$ at a rate of 3 bits per sample.

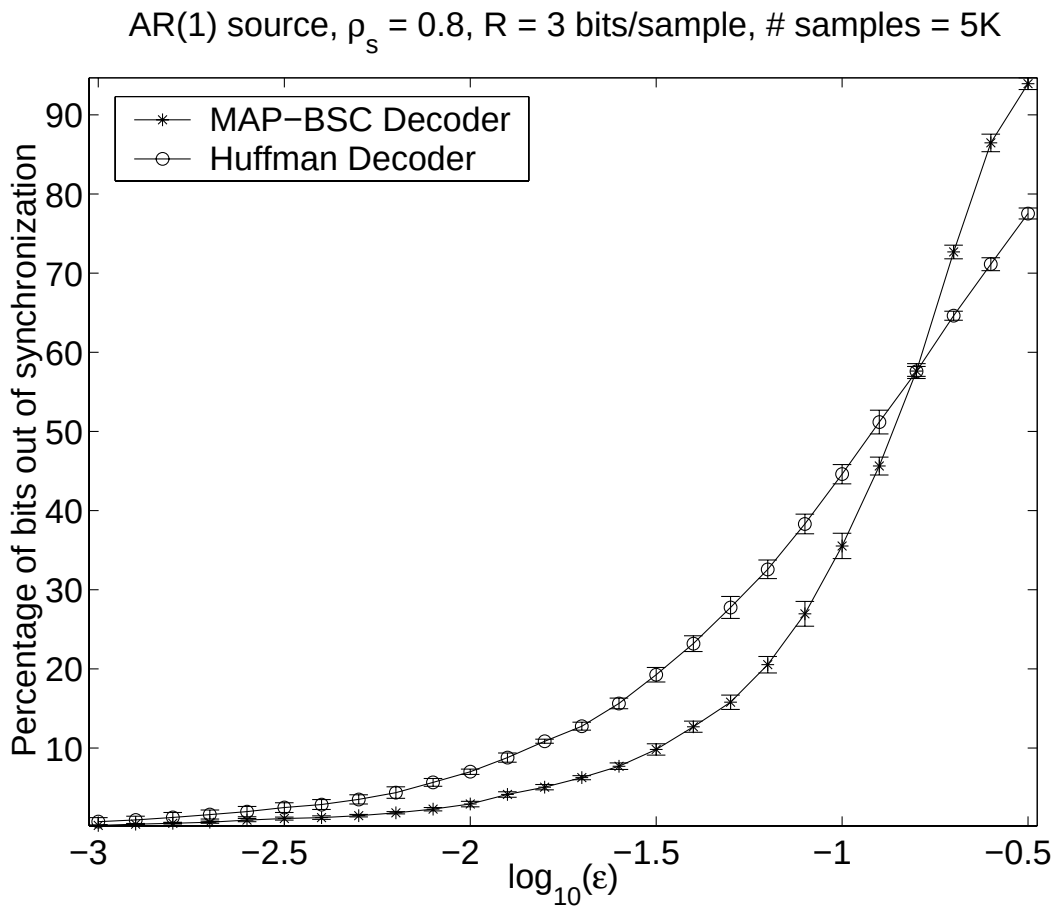


Figure 3.8: Comparison of the percentage loss of synchronization of MAP decoded sequence with that of the Huffman decoded sequence for an $\mathbf{AR}(1)$ source with $\rho_s = 0.8$, quantized to $N = 9$ at a rate of 3 bits per sample.

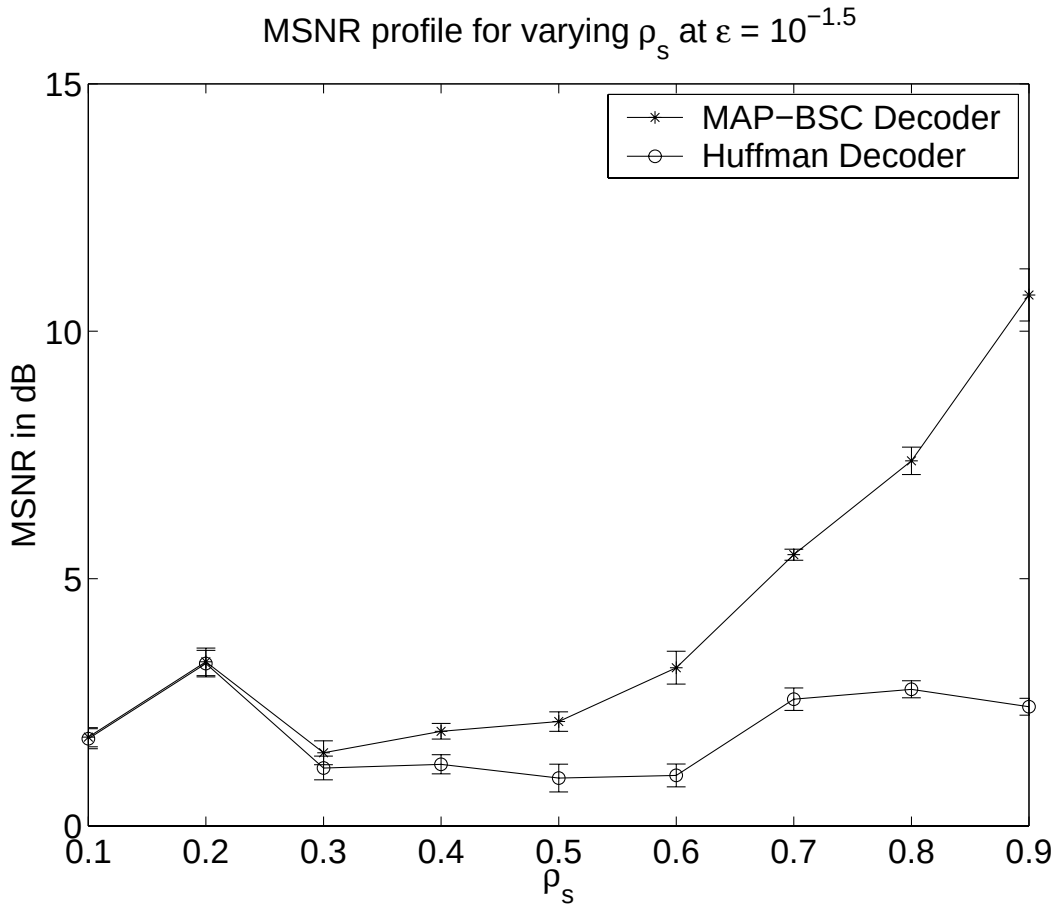


Figure 3.9: MSNR Comparisons for varying ρ_s at $\epsilon = 10^{-1.5}$

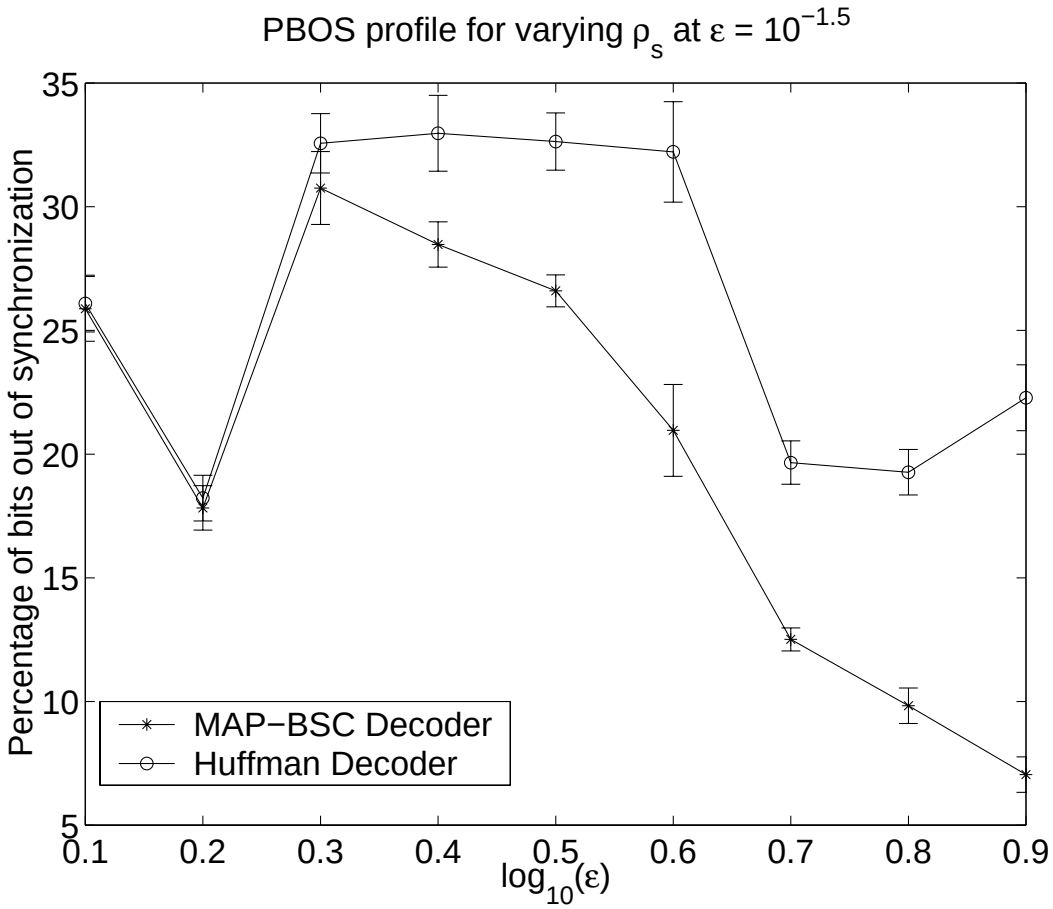


Figure 3.10: PBOS Comparisons for varying ρ_s at $\epsilon = 10^{-1.5}$

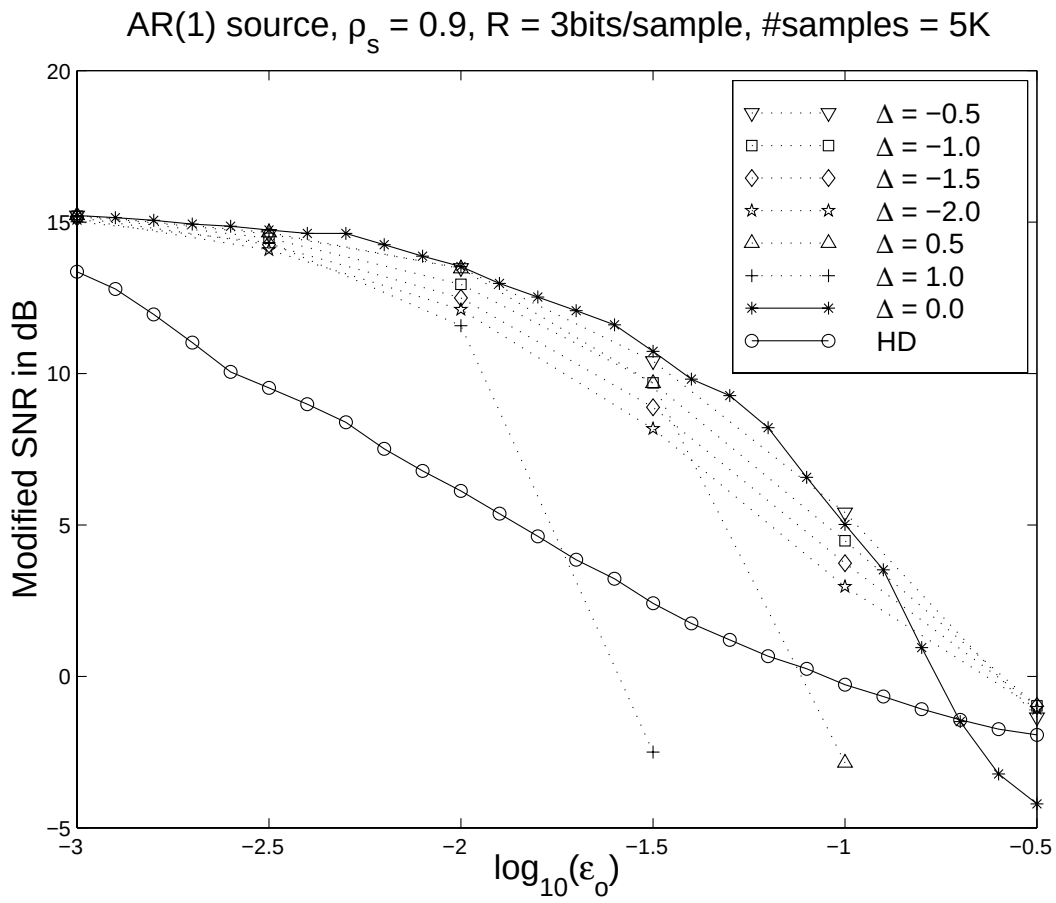


Figure 3.11: Performance of the MAP-BSC decoder under channel mismatch conditions for an $\mathbf{AR}(1)$ source with $\rho_s = 0.9$, quantized to $N = 9$.

channel error mismatch conditions, in terms of the modified SNR and PBOS respectively for the same AR(1) source, with $\rho_s = 0.9$, as used in the previous subsection. The graphs are a plot of the performance measure against the logarithm of the original channel error rate, ϵ_o.

It can be seen from the graphs that the MAP-BSC is quite robust to errors in estimating the channel statistics. As long as the estimated error rate remains less than 2-orders of magnitude lower than the actual error rate, the performance degradation is well below 3 dB. We also note that despite such severe mismatch conditions, the MAP-BSC still out does the Huffman decoder for most error rates. It is interesting to note that it is much better to under-estimate ϵ_o than to over-estimate it; the penalty increases dramatically after ϵ_o increases beyond a threshold value. This observation would seem to indicate that it is better for the decoder to ignore a few errors (under-estimation) than it is for it to "correct" non existing errors as it does when the error rate is over-estimated.

We also note that at very high error-rates, the MAP-BSC operating under a negative Δ mismatch does better than both the $\Delta = 0$ (MAP-BSC) and Huffman decoders. It is thus potentially possible to remove the "cross-over" problem observed in Fig. 3.5 with a judicious choice of Δ! We do not yet have an explanation for this phenomenon; however, we would like to reiterate that the MAP-BSC decoder does not directly optimize either of the performance measures considered here.

Execution Time

Algorithm execution times were measured for our implementations of both the MAP-BSC and Huffman decoders on a SUN ULTRA 10 computer. For a 5000-symbol, $\rho_s = 0.9$ source, using a 9-word codebook, the MAP-BSC method required 6 seconds to complete, compared to 0.38 seconds for the vanilla Huffman decoder. We note, however, that these numbers are only approximate, since neither decoders were fully optimized for speed. Note that the Huffman decoder was implemented using a tree-traversal procedure, which is faster than the table look-up approach.

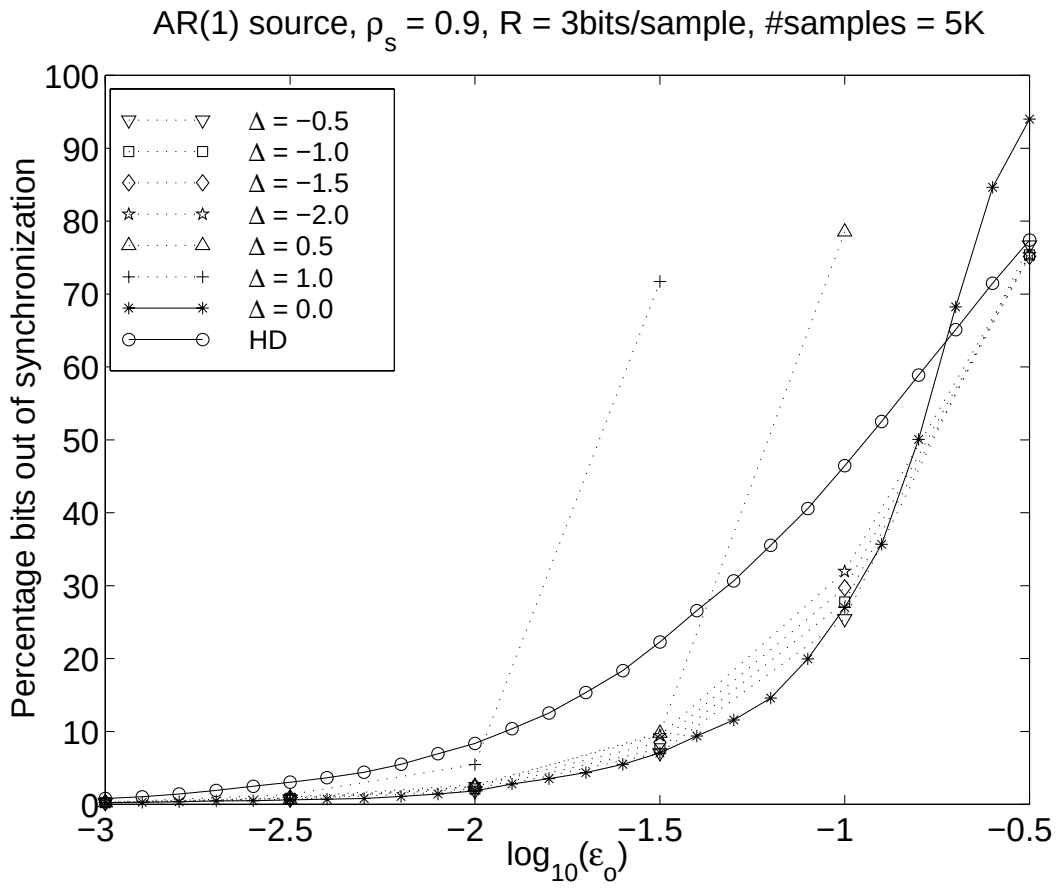


Figure 3.12: Synchronization loss profile under mismatch conditions for an $\mathbf{AR}(1)$ source with $\rho_s = 0.9$, quantized to $N = 9$

3.5.2 MAP-BSC for Memoryless Sources

To study the performance of the MAP-BSC decoder for a memoryless source, 5000 samples of a zero-mean, unit-variance, Gaussian source was quantized with a $N = 9$ level uniform quantizer with step size 0.55 at a rate of 3 bits per sample. The result was then entropy coded and transmitted over a binary symmetric channel. The channel error rates were varied between $10^{-0.5}$ to $10^{-3.0}$ and the performance was averaged over six channel realizations. As before, the quantizer was picked to give good R-D performance under noiseless conditions. The programs used to implement the codec were mostly same as in the case of the AR(1) source, except that the training and the transmitting programs generated Gaussian sources and the MAP decoder program implemented the memoryless-source version of the MAP-BSC algorithm.

Figures 3.13 and 3.14 show the performance of the MAP decoder in comparison with the Huffman decoder. It is seen that the MAP decoder does not offer significant performance improvement over the Huffman decoder. This observation can be explained by the fact that there is no residual redundancy in the form of memory in the source and that the pdf of the source is not peaky enough to provide sufficient residual redundancy in the form of non-uniformity of pdf. The data is also consistent with our observation that the performance advantage of the MAP-BSC decoder decreases with the source memory, as shown in Fig. 3.9.

3.6 Chapter Summary

In summary, this chapter presented the formal statement of the MAP problem for variable-length encoded sources with and without memory, transmitted over binary symmetric channels. A new state-space structure was proposed for the MAP decoder for Markov sources and this was particularized to the case of sources without memory. The MAP algorithm was then presented for both cases and an argument was outlined to show that the proposed algorithms indeed find the exact MAP sequence. A generalization of the MAP problem formulation for Markov sources for higher orders was also provided. It was demonstrated experimentally that the MAP-BSC decoder does significantly better than the Huffman decoder for sources with memory at almost all error rates considered both in terms of the MSNR and the PBOS. It was also shown

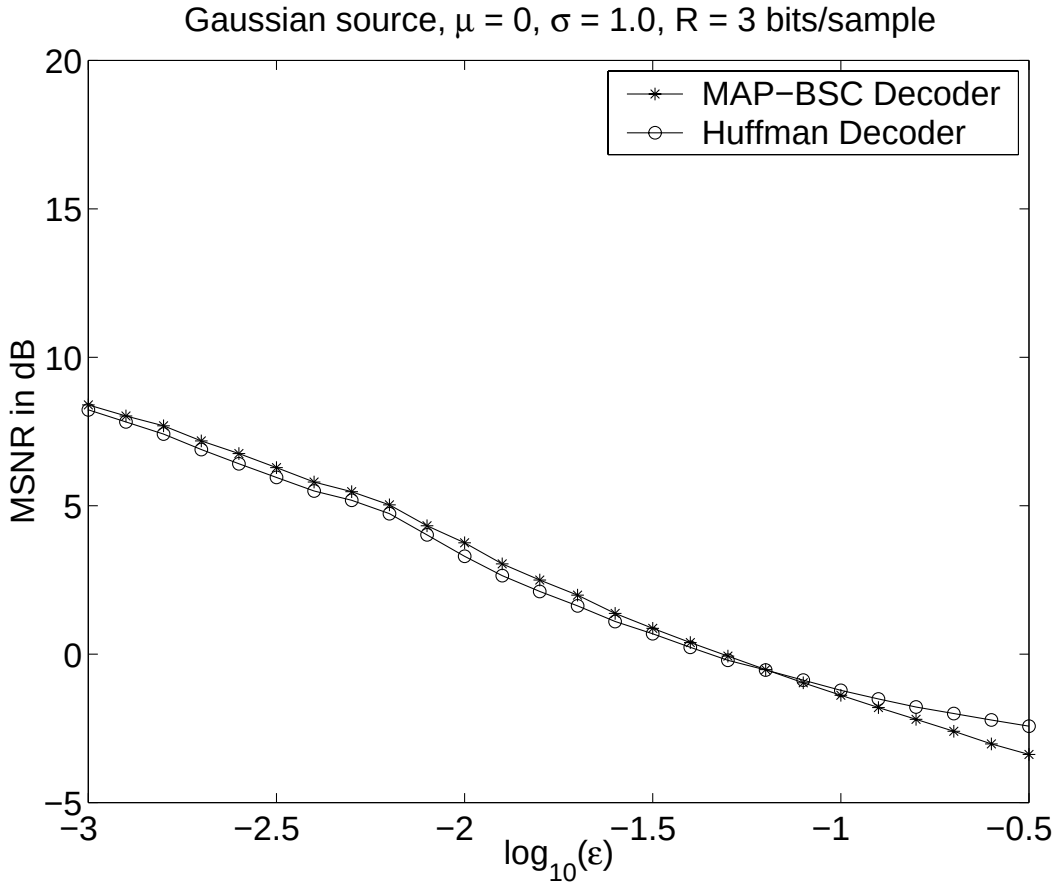


Figure 3.13: Comparison of the modified SNR of the MAP-BSC with the HD for a zero mean, unit variance, Gaussian source quantized to $N = 9$ levels at a rate of 3 bits per sample.

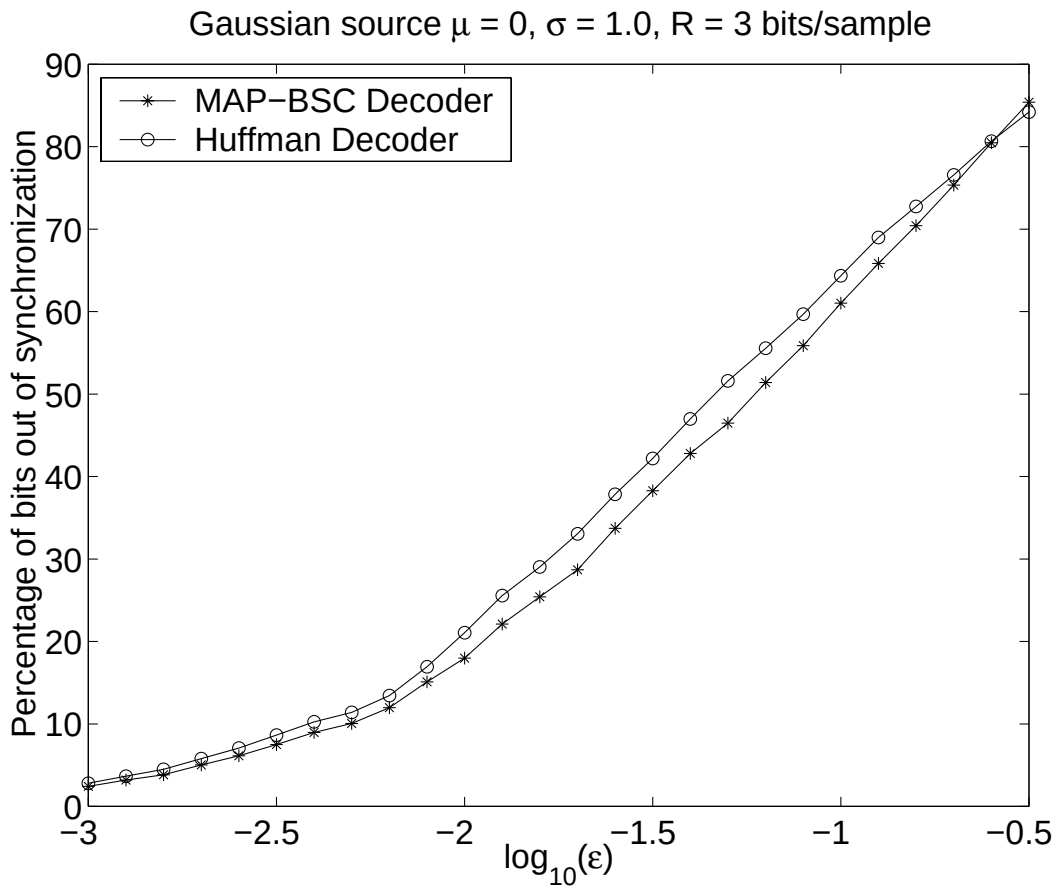


Figure 3.14: Comparison of the percentage loss of synchronization of MAP decoded sequence with that of the Huffman decoded sequence for a zero mean, unit variance, source quantized to $N = 9$ levels at a rate of 3 bits per sample.

that the residual redundancy in the source decreased with source correlation, leading to a decrease in the performance improvement over the Huffman decoder. Finally, it was shown experimentally that for the memoryless source considered, the MAP-BSC does not perform much differently from the Huffman decoder. Testing the algorithm for memoryless sources with peakier distribution is left for future work. Performance of the MAP-BSC decoder was studied under channel mismatch conditions and it was shown that the MAP-BSC decoder is robust to inaccuracies in the estimation of channel statistics. Finally, we close this chapter with the remark that the proposed algorithms are equally applicable to *any* entropy code that can be implemented as a table look-up algorithm, even though we presented results for the more commonly used Huffman codes.

Chapter 4

MAP Decoding: Additive Markov Channels

In this chapter, we consider the design of an optimal joint source-channel decoder for variable-length encoded sources over channels with memory. This work is motivated by the facts that most real-world channels are not memoryless and that (variable-length) entropy coding is generally required for a source-coder to achieve good rate-distortion performance. Although interleaving can be used to effectively convert a bursty channel to a BSC, this strategy also results in delays, which may be unacceptable in some applications. Hence, it is also of interest to design a MAP decoder for channels with memory. In this chapter, we will describe one of the common models for a channel with memory and develop the MAP decoder for entropy coded Markov sources transmitted over this channel. We will also study the performance of this decoder under mismatch conditions which will demonstrate the robustness of the decoder.

4.1 The Coding Framework

Consider consecutive samples of a continuous, stationary, entropy coded (eg. Huffman coded) Markov source are transmitted over an *additive Markov channel* (AMC). This channel can be described by the equation

$$Y_i = X_i \oplus Z_i, \quad \forall\, i = 1, 2, \ldots \tag{4.1}$$

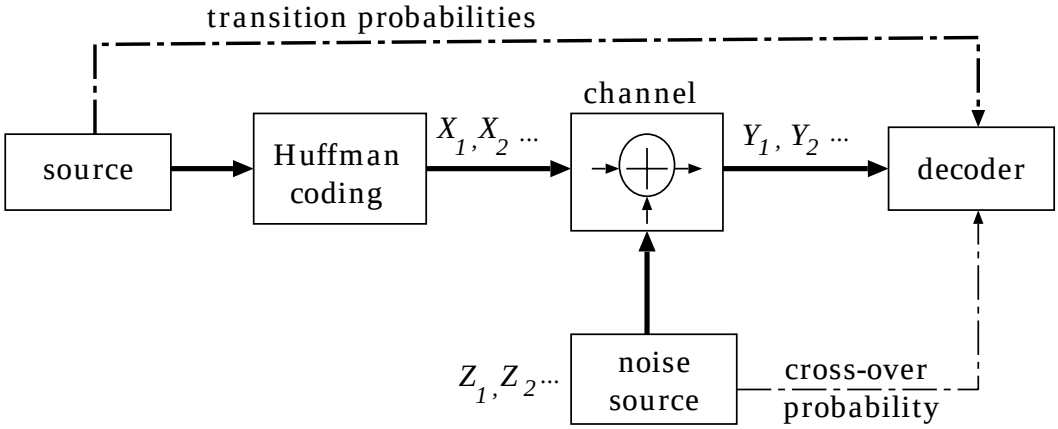


Figure 4.1: System Block Diagram

where X_i, Z_i and Y_i are binary and represent the input, the noise and the channel output bit, respectively and $\oplus$ denotes the binary XOR operation The noise process $\{Z_i\}$ is assumed to be a stationary, ergodic, Markov process, uncorrelated with the input. For a channel with error probability ϵ, and correlation coefficient ρ_c, the transition probabilities $Q(z_n|z_{n-1}) \equiv \Pr\{Z_n = z_n|Z_{n-1} = z_{n-1}\}$ and the marginal probabilities of the noise bits $Q(z_n) \equiv \Pr\{Z_n = z_n\}$ are given by (Burlina, Alajaji, and Chellappa 1996):

$$\begin{bmatrix} Q(0|0) & Q(1|0) \\ Q(0|1) & Q(1|1) \end{bmatrix} = \frac{1}{1+\delta} \begin{bmatrix} 1-\epsilon+\delta & \epsilon \\ 1-\epsilon & \epsilon+\delta \end{bmatrix},$$

with $Q(1) = \epsilon = 1 - Q(0)$ and where $\delta = \rho_c/(1-\rho_c)$ is a correlation parameter. A block diagram showing how the noise is added is presented in Fig. 4.1.

4.2 The MAP-AMC Problem and the State Space

As in the previous chapter, let $\mathbf{c}_j^{n(j)}$ and $\mathbf{r}_j^{n(j)}$ represent the transmitted and received codeword sequences under the j^{th} partition, where $n(j)$ is the number of codewords in the stream. The problem is to find the most probable transmitted sequence given the

received sequence, the source statistics and the channel statistics. Hence, if $\hat{j}$ represents the index of the most probable transmitted sequence, then the MAP problem for the AMC (MAP-AMC) is to determine the optimal sequence $\mathbf{c}_{\hat{j}}^{n(\hat{j})} = \{c_{j,1}, \ldots, c_{j,n(\hat{j})}\}$ according to:

$$\mathbf{c}_{\hat{j}}^{n(\hat{j})} = \arg \max_{\mathbf{c}_j^{n(j)}} \left\{ \Pr(c_{j,1}) \epsilon^{Z_{j,1}} (1-\epsilon)^{(1-Z_{j,1})} \prod_{k=2}^{n(j)} [\Pr(c_{j,k}|c_{j,k-1})] \prod_{i=2}^{B} Q(Z_{j,i}|Z_{j,i-1}) \right\} \quad (4.2)$$

where $\Pr(c_{j,1})$ is the probability that codeword $c_{j,1}$ was transmitted first, $\Pr(c_{j,k}|c_{j,k-1})$ is the probability that the codeword $c_{j,k}$ was transmitted immediately after the codeword $c_{j,k-1}$ and $Z_{j,i}$, is the i^{th} noise bit under the j^{th} partition, which is determined by the received bit stream, the specific partition and Eqn 4.1. This maximization over j, effectively searches through all possible error sequences and bit stream partitions. Eqn 4.2 and Eqn 3.1 differ in the channel probability term only. For the MAP-BSC (Eqn 3.1), this term reduces to one based on the Hamming distance between the received and transmitted sequences, because the channel is memoryless and hence, only the *number* of bit errors matters, not the *positions* in which they occur.

Before we go on to state the MAP-AMC algorithm, we need to look at the state-space that can be used in the dynamic program associated with this problem. To this end, we consider the metric that needs to be maximized, namely, the right hand side of Eqn 4.2. This metric can be factored into two terms: a channel metric term and a source metric term, the former being

$$\Pr(c_{j,1}) \epsilon^{Z_{j,1}} (1-\epsilon)^{(1-Z_{j,1})} \prod_{i=2}^{B} Q(Z_{j,i}|Z_{j,i-1}) \quad .$$

During the dynamic program, every stage must know the noise-bit of the previous stage in order to compute the continued-product occurring in the channel term. This requirement translates into remembering the noise bit associated with the last bit of the codeword that immediately precedes the current node (state-stage pair), along the best metric path terminating at the current node. This task can be accomplished within the frame-work of the state-space used for the MAP-BSC and hence we may use the state-space developed in Chapter 3 for the MAP-AMC.

4.3 MAP-AMC Decoding Algorithm

The MAP-AMC decoding algorithm performs *three* main operations at each stage. Two of them are the same as in the MAP-BSC case: one consists of examining the metrics of all the paths entering each node in the trellis and the other consists of looking for a merger of the paths and the actual declaration of decoded codewords (if there is a merge). The third operation involves remembering the noise bit associated with predecessor nodes. For a complete node, the algorithm must remember whether the last bit of the predecessor codeword, along the maximum-metric path, was noisy or not. For incomplete nodes, however, a vector of values needs to be remembered corresponding to the last bits of *each* of the complete nodes of the previous stage.

Since the metric that is being maximized is different from the one for the MAP-BSC for Markov sources, the metric increment ($\mathcal{M}[k,i,c_j]$) associated with the path segment that begins at node $(j,(i-d_k))$ and terminates in the node (k,i); for complete states k and j, is different than for the BSC. We need to define some additional notation before we can give an expression for the metric increment. As before, let $\mathbf{b}_{k,i}$ represent the d_k most recently received bits at stage i. Now, let $\mathbf{b}_{k,i}(a)$, denote the a^{th} bit in this segment of bits. Similarly, let $c_k(a)$ denote the a^{th} bit of the d_k dimensional codeword c_k, associated with the state k. Now, we define $\mathcal{M}[k,i,c_j]$ as follows

$$\mathcal{M}[k,i,c_j] \;=\; \log_{10}(\Pr\{c_k|c_j\}) + \sum_{a=0}^{d_k-1} \log_{10} Q(z_{k,(i-a)}|z_{k,(i-a-1)}) \;\;, \qquad (4.3)$$

where $z_{k,(i-a)} = \mathbf{b}_{k,i}(d_k-a) \oplus c_k(d_k-i)$ for $a \in \{0,\cdots,d_k-1\}$ and $z_{k,i-d_k} = \mathcal{N}_{N+d_k-1}(j)$, which is 1 if the last bit of the codeword c_j was different from the corresponding received bit and 0 otherwise. We note that a is merely a running index and has no other significance. A more formal expression for $\mathcal{N}$ will be presented when stating the algorithm. Eqn 4.3 differs from the expression for the metric increment of the MAP-BSC (Eqn 3.2) in that it is no longer possible to express the channel probability term of the metric as a term based on the overall Hamming distance between the received sequence and the transmitted sequence, as explained earlier.

For the incomplete states, we define $M_{k,i}(u)$ just as in the case of the MAP-BSC, to be the u^{th} element in the vector of metric values that need to be remembered at (k,i), where $k \in \{N+1,\ N+2,\cdots,N+l_{\max}-1\}$ and $u \in \{1,2,\cdots,N\}$. The

algorithm can then be formally stated as

Initialize:

Input first $l_{\min} - 1$ bits

For $i = 1, \cdots, l_{\min} - 1$,

$M_{k,i} \leftarrow 0 \ \forall k \in \{1, 2, \cdots, N\}$.

$M_{k,i}(u) \leftarrow 0 \ \forall \ u \in \{1, 2, \cdots, N\}; \forall k \in \{N+1, \cdots, N + l_{\max} - 1\}$.

$i \leftarrow l_{\min}$

For $k = 1, 2, \cdots, N$

$\mathbf{v} = (k, i)$

$\mathbf{v}^p = \Phi$ (no parents)

Input:

Input bit at the i^{th} stage

Update path metrics:

For $k = 1, 2, \cdots, N$

$$M_{k,i} \leftarrow \begin{cases} \max_j \{M_{(N+d_k-1),(i-1)}(j) \ + \ \mathcal{M}[k, i, c_j]\}, & i > d_k \\ \log_{10}(\Pr\{c_k\}) + z_{k,1} \log_{10}(\epsilon) + (1 - z_{k,1}) \log_{10}(1 - \epsilon) & \\ \qquad + \sum_{b=0}^{d_k - 2} \log_{10} Q(z_{k,i-b} | z_{k,i-b-1}), & i = d_k \\ 0 \ , \quad i < d_k & \end{cases}$$

$$M_{k,i}(u) \leftarrow \begin{cases} M_{u,(i-1)} & \forall \ u \ \in \{1, 2, \cdots, N\}, \ k = N + 1. \\ M_{(k-1),(i-1)}(u) & \forall \ u \ \in \{1, 2, \cdots, N\}, \ \forall k \in \{N+2, \cdots, N + l_{\max} - 1\} \end{cases}$$

$$j_k^* \leftarrow \arg \max_j \{M_{(N+d_k-1),(i-1)}(j) + \mathcal{M}[k, i, c_j]\}, \ \forall k \ \in \{1, 2, \cdots, N\}.$$

As before, updating $M_{k,i}(u)$ involves only a copy operation whereas updating $M_{k,i}$ involves some calculations.

Update noise-bits:

For $k = 1, 2, \cdots, N$

$$\mathcal{N}_{k,i} \leftarrow \begin{cases} \mathbf{b}_{k,i}(d_k) \oplus c_k(d_k),\ i \geq d_k \\ \text{Don't care}, \quad i < d_k \end{cases}$$

$$\mathcal{N}_{k,i}(u) \leftarrow \begin{cases} \mathcal{N}_{u,(i-1)} & \forall\ u\ \in \{1,2,\cdots,N\},\ k = N+1. \\ \mathcal{N}_{(k-1),(i-1)}(u) & \forall\ u\ \in \{1,2,\cdots,N\},\ \forall k \in \{N+2,\cdots,N+l_{\max}-1\} \end{cases}$$

Update paths:

For $k \in \{1,2,\cdots,N\}$

$\mathbf{v} = (k,i)$

$v_x^p = j_k^*;\ v_y^p = i - d_{j_k^*};$

Merge Check and Output:

Merge declaration is same as in the MAP-BSC. As before, we define a set of nodes $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_{Nl_{\max}}\}$ and initialize them to the nodes from which we need to trace back in order to determine a merge. We need $Nl_{\max}$ such nodes, since each degree of incompleteness contributes to N nodes and the N complete nodes contribute to one parent each. $\mathbf{V}$ is formed according to

$\mathbf{v}_m = (m - N\lfloor \frac{m}{N} \rfloor,\ i - \lfloor \frac{m}{N} \rfloor),\ \forall m \in \{1,2,\cdots,Nl_{\max}\}.$

If $(i \geq l_{\max}+1)$

a: *If* (the parents of all the nodes in $\mathbf{V}$ are equal)

decide that the path terminating in $\mathbf{v}_1$ was indeed transmitted

and set the parent of all the nodes at this stage to be Φ.

Else

$\mathbf{v}_m \leftarrow \mathbf{v}_m^p,\ \forall\ m\ \in \{1,2,\cdots,Nl_{\max}\}$

If $(\mathbf{v}_m^p \neq \Phi\ \ \forall\ m\ \{1,2,\cdots,Nl_{\max}\})$

go to **a**.

If (end of sequence)

$k^*\ \leftarrow\ \arg\ \max_{k\in\{1,2,\cdots,N\}} M_{k,i}$

Declare the path terminating at (k^*,i) as the optimal path.

Else

$i\ \leftarrow\ i+1$

go to **Input**.

Similar arguments to those outlined in the previous chapter may be used to show that the algorithm presented in this chapter does indeed find the MAP sequence. Finally, a remark on the complexity of the MAP-AMC algorithm. The MAP-AMC algorithm uses the same state-space as the MAP-BSC for Markov sources; however, there is one additional operation at each stage in the MAP-AMC, namely, remembering the noise bit corresponding to the latest stage. Hence the complexity can be considered to be slightly more than the MAP-BSC algorithm. Just as in the MAP-BSC case, the complexity can be further reduced when the greatest common divisor of the lengths of the codewords is greater than one.

Similar arguments to those made in the previous chapter can also be extended to the MAP-AMC to show how the MAP-AMC presented for a Markov source of degree one can be extended to Markov sources of higher orders.

4.4 Experimental Results

The performance of the MAP-AMC decoder and the Huffman decoder are evaluated using the same performance measures as described before. Experiments were performed on $50,000$ samples of zero mean, first order, Gauss-Markov sources (driven by a unit variance, Gaussian random process) with $\rho_s = 0.9$ and $\rho_s = 0.8$ quantized using the quantizers described in the previous chapter. Hence the sources are the same as in the previous chapter, only the number of samples is higher. This increase was necessary, in order to generate the noise process (it's correlation and the required error rate) accurately. The quantized source was then Huffman encoded and corrupted by random bit error patterns representing the AMC. The results presented here are an average of six channel realizations and the range of values resulting from the different seeds is shown in the figures as error-bars.

Just as in the MAP-BSC case (Section 3.5.1), four programs were used to implement the codec. The training and the transmitting programs were the same as in Section 3.5.1, but the channel program was different in that it implemented the AMC as per Equation 4.1 and the input to this program were the channel error rate, the channel correlation coefficient and the transmitted stream. Different channel realizations could be simulated using different random number seeds. The MAP decoder program implemented the MAP-AMC version of the algorithm. The inputs to this

program were the corrupted stream, the value of the channel cross-over probability, the channel correlation coefficient, the files corresponding to the Huffman codes and the source statistics. The Huffman decoder is the same as before.

The performance of the MAP-AMC is compared to that of the Huffman decoder using the MSNR and PBOS as in the previous chapter. Figure 4.2 shows the decoded MSNR for both the MAP-AMC and the Huffman decoder for the AR(1) source with correlation coefficient $\rho_s = 0.9$. It is seen that the MAP-AMC performs better than the Huffman decoder for all the error rates considered. The performance improvement in MSNR is just under 1dB at very low error rates ($10^{-3.0}$) and increases until it reaches a peak of about 6.31 dB at an error rate of about $10^{-1.2}$ and then starts to drop until it reaches 4.72 dB at an error rate of $10^{-0.5}$.

Fig. 4.3 is a plot of the average PBOS in the decoded sequence for both the MAP-AMC and the Huffman decoder for the same source and channel as described above. It is seen from this plot that the MAP-AMC performs better than the Huffman decoder in terms of the PBOS, for all the error rates considered; however, the pattern of improvement is slightly different from the MSNR curve. Here, the performance improvement at very low error rates, $10^{-3.0}$, is almost zero and keeps increasing with error rate until it reaches a maximum of 27.29% at $10^{-0.5}$.

Figures 4.4 and 4.5 are plots of MSNR and PBOS, respectively, for the AR(1) source with $\rho_s = 0.8$, transmitted over the same AMC as above. We note that the performance trend is the same as in the source with $\rho_s = 0.9$, with the maximum values of MSNR and PBOS being 3.54 dB (at $10^{-1.1}$) and 16.59% (at $10^{-0.5}$) respectively. Hence we see that, just as in the case of the MAP-BSC, the performance improvement that the MAP-AMC offers over the Huffman decoder reduces as the source memory reduces, all other conditions remaining the same.

We then looked at the effect of reducing the channel memory by returning to the $\rho_s = 0.9$ source, but reducing ρ_c to 0.7. Fig. 4.6 and Fig. 4.7 give the comparison of the MAP-AMC decoder with the Huffman decoder in terms of the MSNR and PBOS respectively. The performance trend seen in this case is the same as that seen in the $\rho_c = 0.8$ case, with the maximum in MSNR and PBOS being 6.67 dB (at $10^{-1.3}$) and 20.87% ($10^{-0.6}$) respectively. We then set the error rate to $\epsilon = 10^{-1.5}$ and swept ρ_c from $0.1 \rightarrow 0.9$. The performance plots are shown in Fig 4.8 and Fig 4.9 and, interestingly, it is seen that the performance of *both* the MAP-AMC and

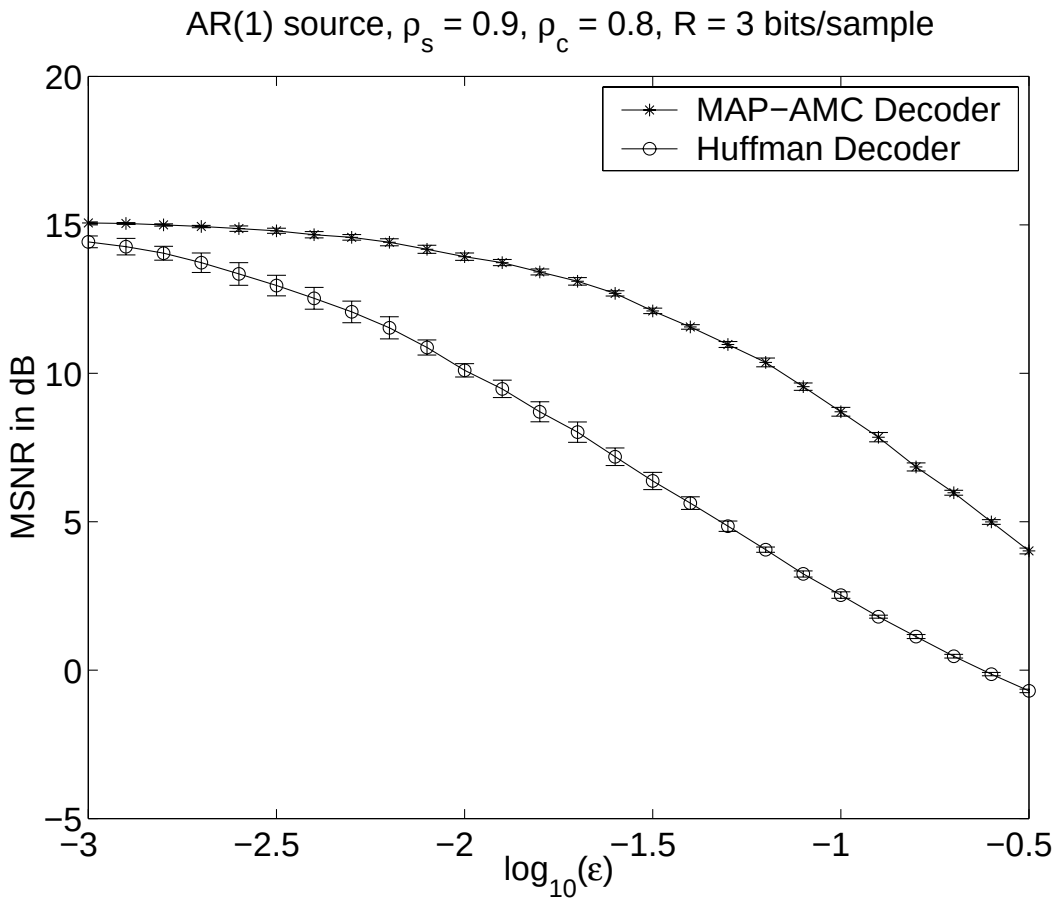


Figure 4.2: Comparison of the MSNR of the MAP-AMC with the HD for the AR(1) source with $\rho_s = 0.9$, quantized using an $N = 9$ level uniform quantizer with step size 1.25 at a rate of 3 bits per samples, transmitted over an AMC with correlation coefficient $\rho_c = 0.8$.

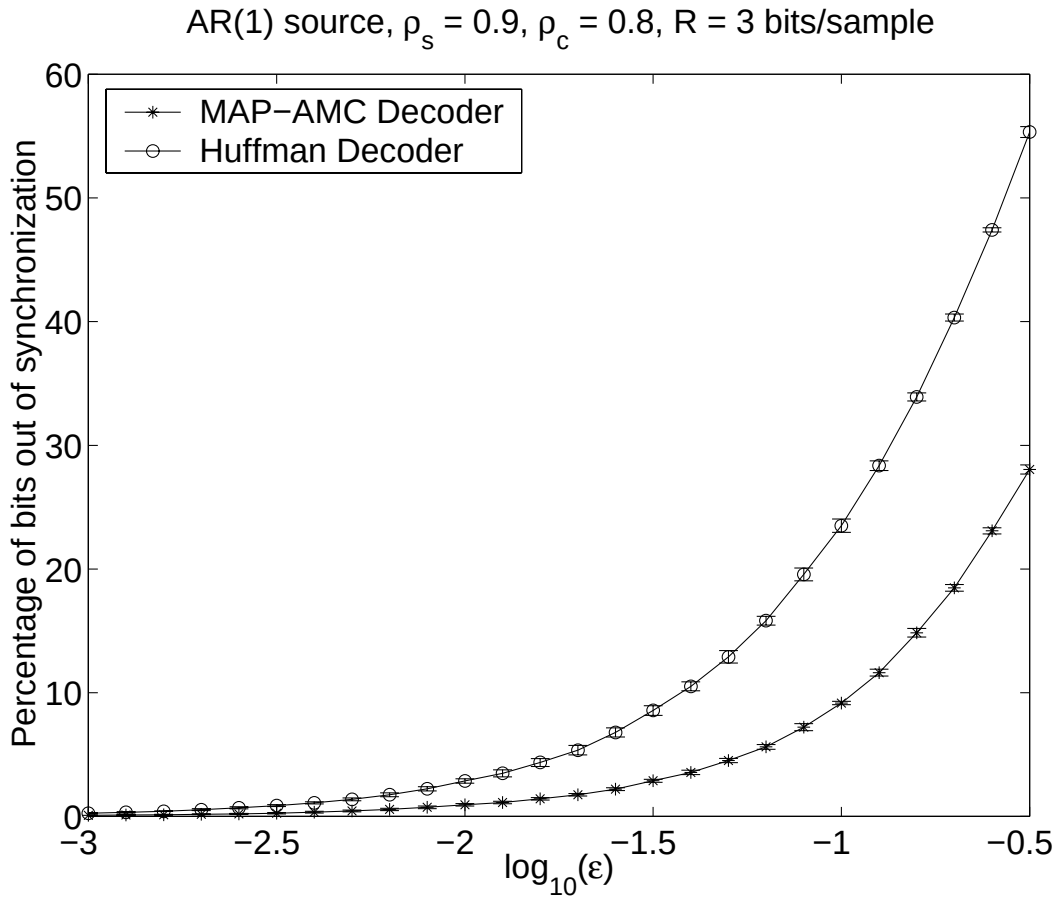


Figure 4.3: Comparison of the percentage loss of synchronization of MAP-AMC decoded sequence with that of the Huffman decoded sequence for an $\mathbf{AR}(1)$ source with $\rho_s = 0.9$, quantized using an $N = 9$ level uniform quantizer with step size 1.25 at a rate of 3 bits per samples, transmitted over an AMC with correlation coefficient $\rho_c = 0.8$.

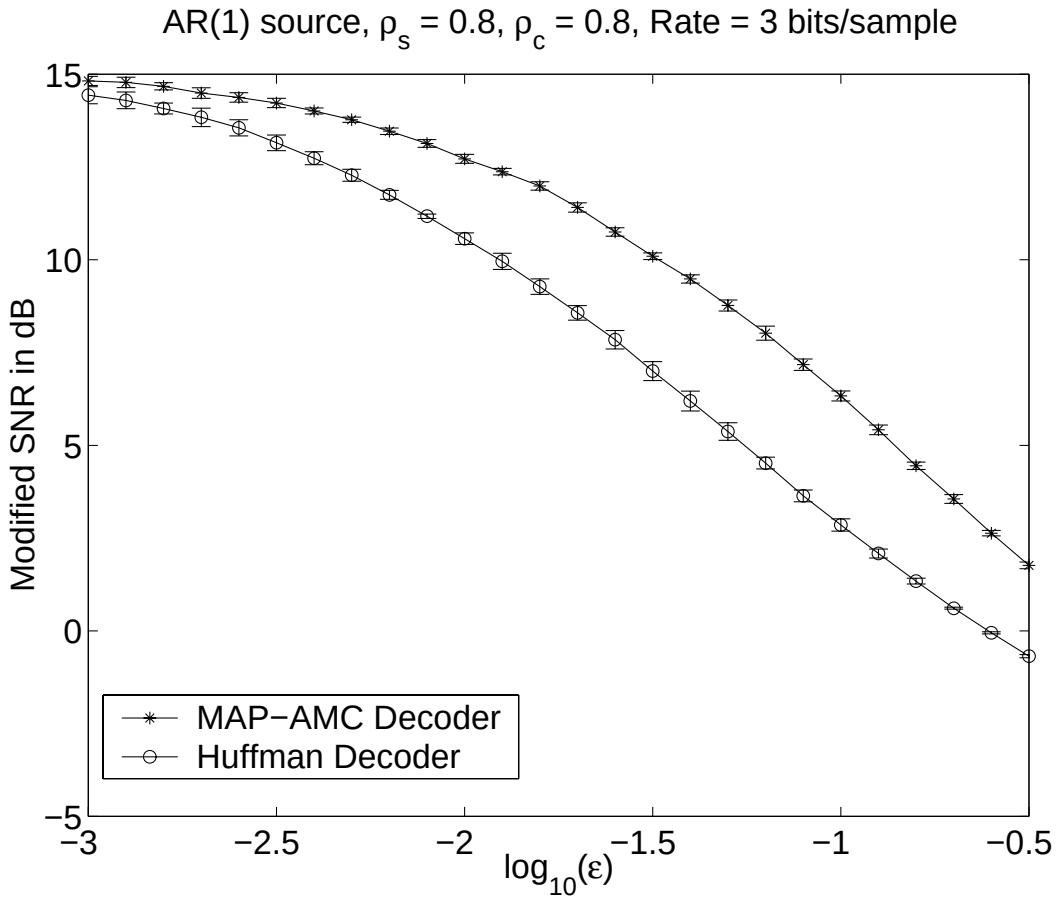


Figure 4.4: Comparison of the percentage loss of synchronization of MAP-AMC decoded sequence with that of the Huffman decoded sequence for the AR(1) source with $\rho_s = 0.8$, quantized using an $N = 9$ level uniform quantizer with step size 0.9 at a rate of 3 bits per samples, transmitted over an AMC with correlation coefficient $\rho_c = 0.8$.

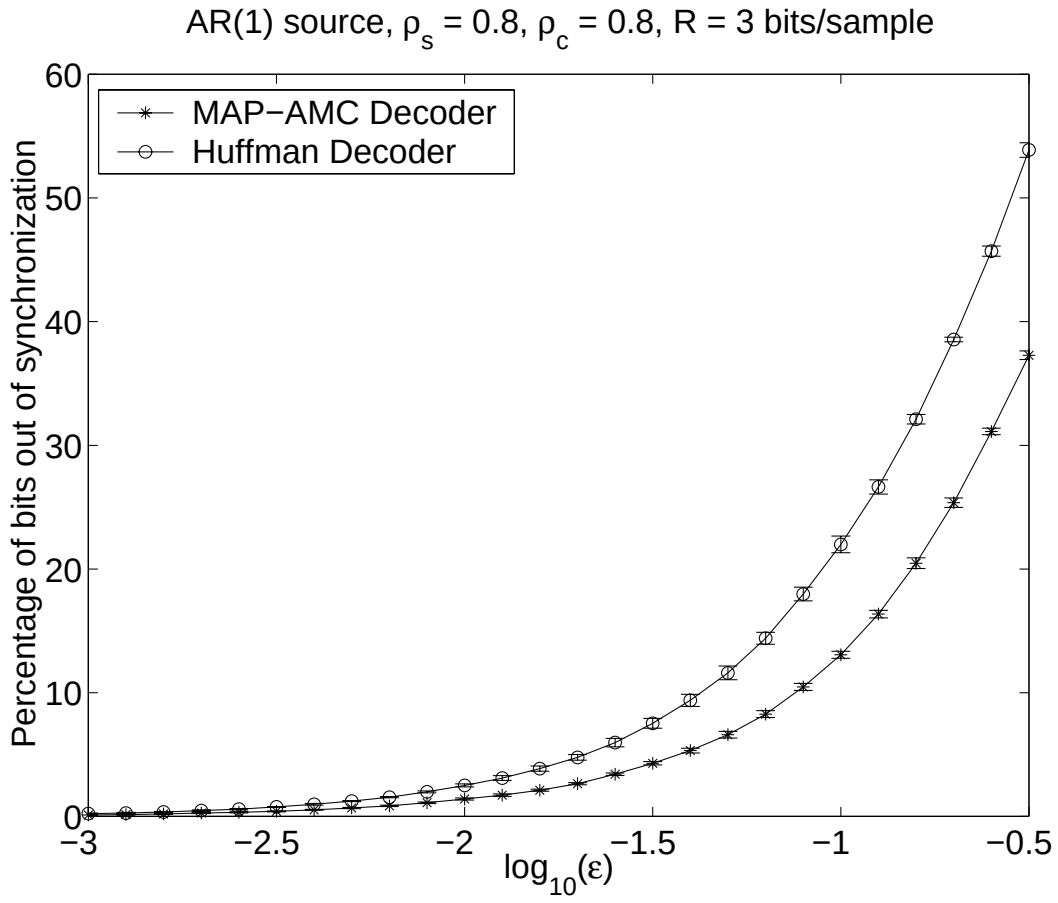


Figure 4.5: Comparison of the percentage loss of synchronization of MAP-AMC decoded sequence with that of the Huffman decoded sequence for an AR(1) source with $\rho_s = 0.8$, quantized using an $N = 9$ level uniform quantizer with step size 0.9 at a rate of 3 bits per samples, transmitted over an AMC with correlation coefficient $\rho_c = 0.8$.

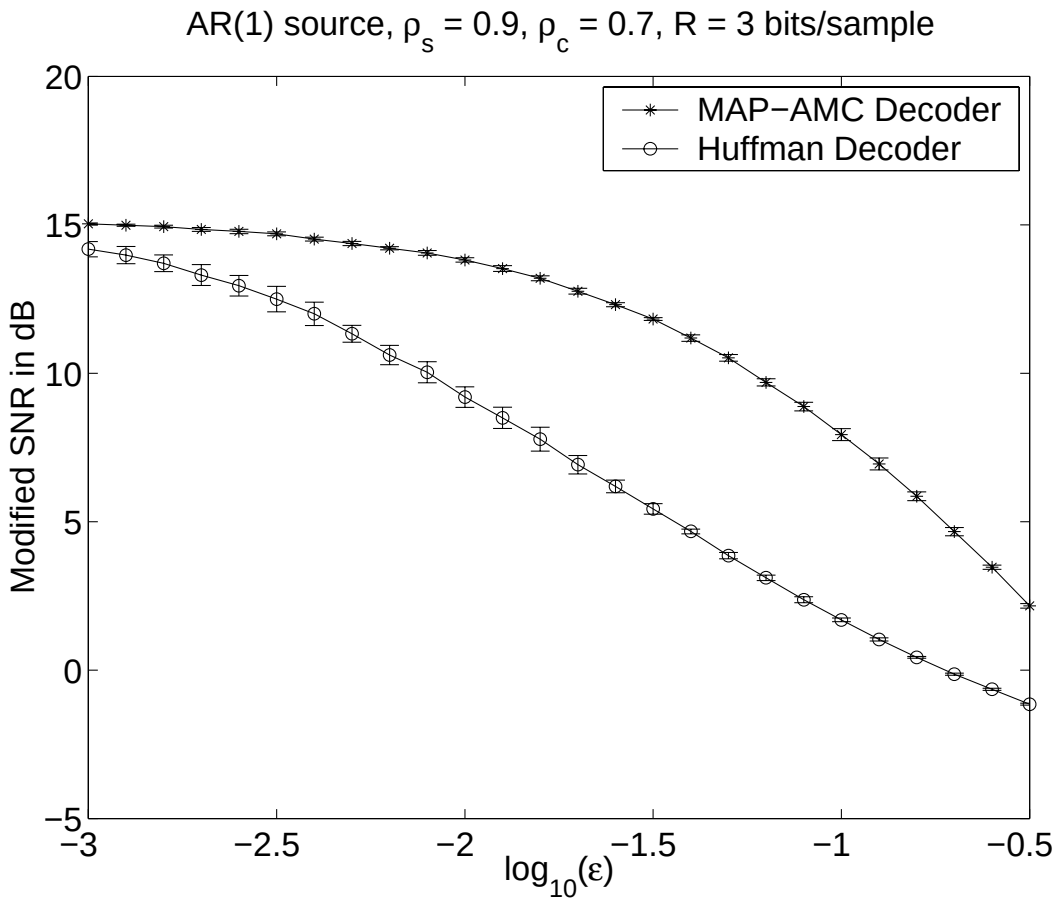


Figure 4.6: Comparison of the MSNR of the MAP-AMC with the HD for an AR(1) source with $\rho_s = 0.9$, quantized using an $N = 9$ level uniform quantizer with step size 1.25 at a rate of 3 bits per samples, transmitted over an AMC with correlation coefficient $\rho_c = 0.7$.

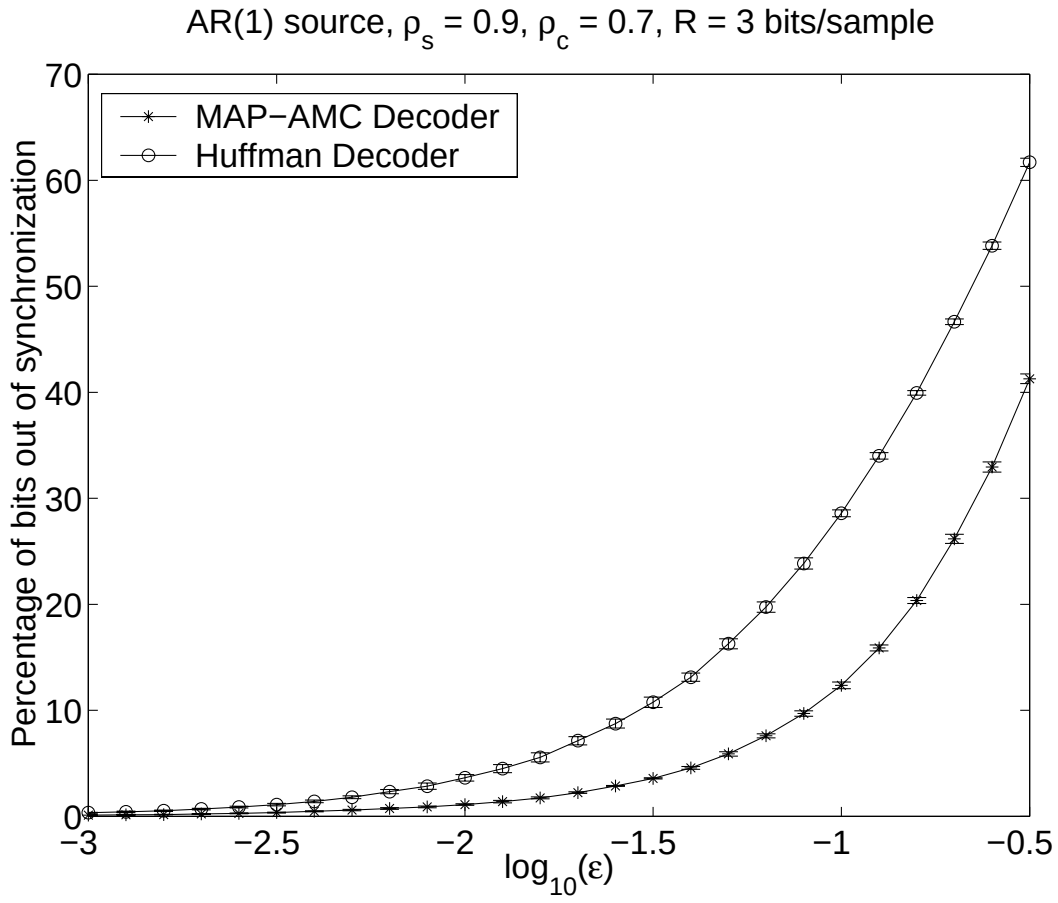


Figure 4.7: Comparison of the percentage loss of synchronization of MAP-AMC decoded sequence with that of the Huffman decoded sequence for an $\mathbf{AR}(1)$ source with $\rho_s = 0.9$, quantized using an $N = 9$ level uniform quantizer with step size 1.25 at a rate of 3 bits per samples, transmitted over an AMC with correlation coefficient $\rho_c = 0.7$.

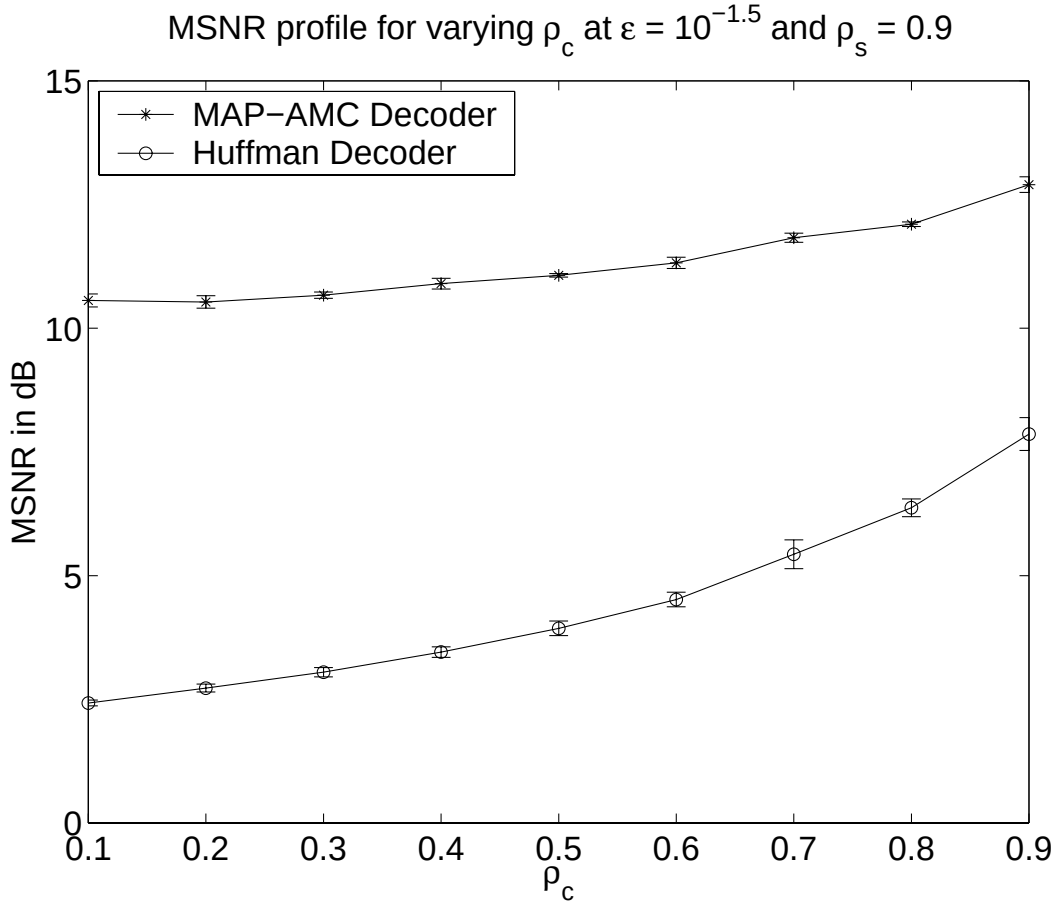


Figure 4.8: Effect of Channel Memory on MSNR: $\rho_s = 0.9$, $\epsilon = 10^{-1.5}$.

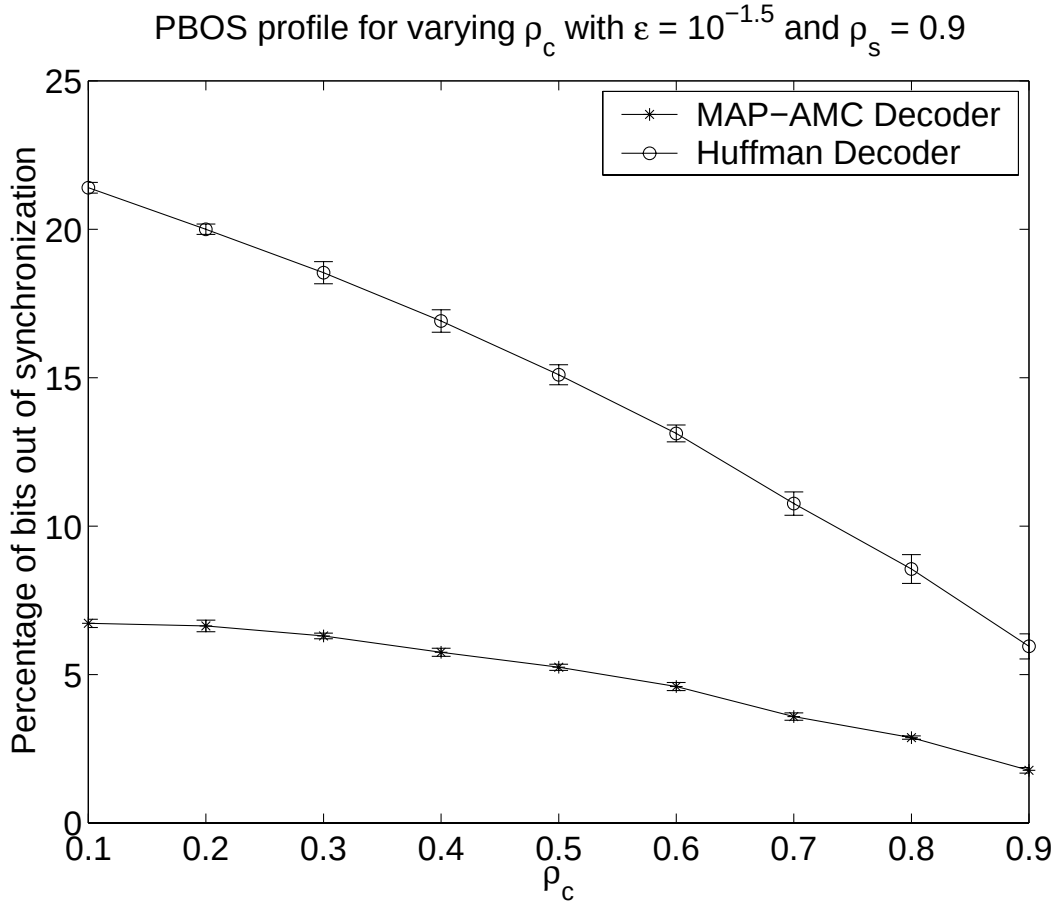


Figure 4.9: Effect of Channel Memory on PBOS: $\rho_s = 0.9$, $\epsilon = 10^{-1.5}$.

Huffman decoders increases with the channel memory. Although somewhat surprising at first thought, the Huffman results can be explained by the fact that increasing the channel memory results in greater error-clustering: the result is fewer symbols being corrupted and a smaller number of synchronization losses. We also note that the relative performance of the MAP-AMC decoder is always better than the Huffman decoder and actually increases as ρ_c is dropped.

We note that setting the source-memory to zero (see Section 4.4.2) results in a MAP-AMC decoder performance that is essentially indistinguishable from that of the Huffman decoder; this result supports the conclusion that channel-memory is not effectively exploited by this decoder when the source is memoryless. However, for large values of source memory, our experiments show that the channel memory does have a positive effect on the performance of the MAP-AMC decoder (see Fig 4.8). It must also be noted that it is difficult to separate the improvement of the MAP-AMC decoder over Huffman decoder into that due to the source memory and that due to the channel memory.

To give an idea of computational complexity, we report that each of the data points shown in Figs. 4.2–4.8 used approximately 1 minute of CPU time on a SUN ULTRA 10 computer for the MAP-AMC decoder and about 4 seconds for the Huffman decoder.

4.4.1 Performance Under Mismatch Conditions

In this section, we consider the performance of the MAP-AMC decoder under channel-correlation mismatch conditions only, as it is reasonable to expect that the performance of the MAP-AMC under channel error rate mismatch conditions will be the same as in the case of the MAP-BSC. As before, $50,000$ samples of a Gauss-Markov source of $\rho_s = 0.9$ were considered. The channel used to transmit this was an AMC of $\rho_c = 0.8$. The decoder was designed using an "erroneous" channel correlation value of ρ_d ranging from 0 to 0.9. The results, averaged over six channel realizations are shown in Fig. 4.10 and 4.11 and it is seen that the MAP-AMC is reasonably robust to errors in estimating ρ_c and only a huge mis-match of 0.9 causes the MAP-AMC result to drop below the Huffman curve.

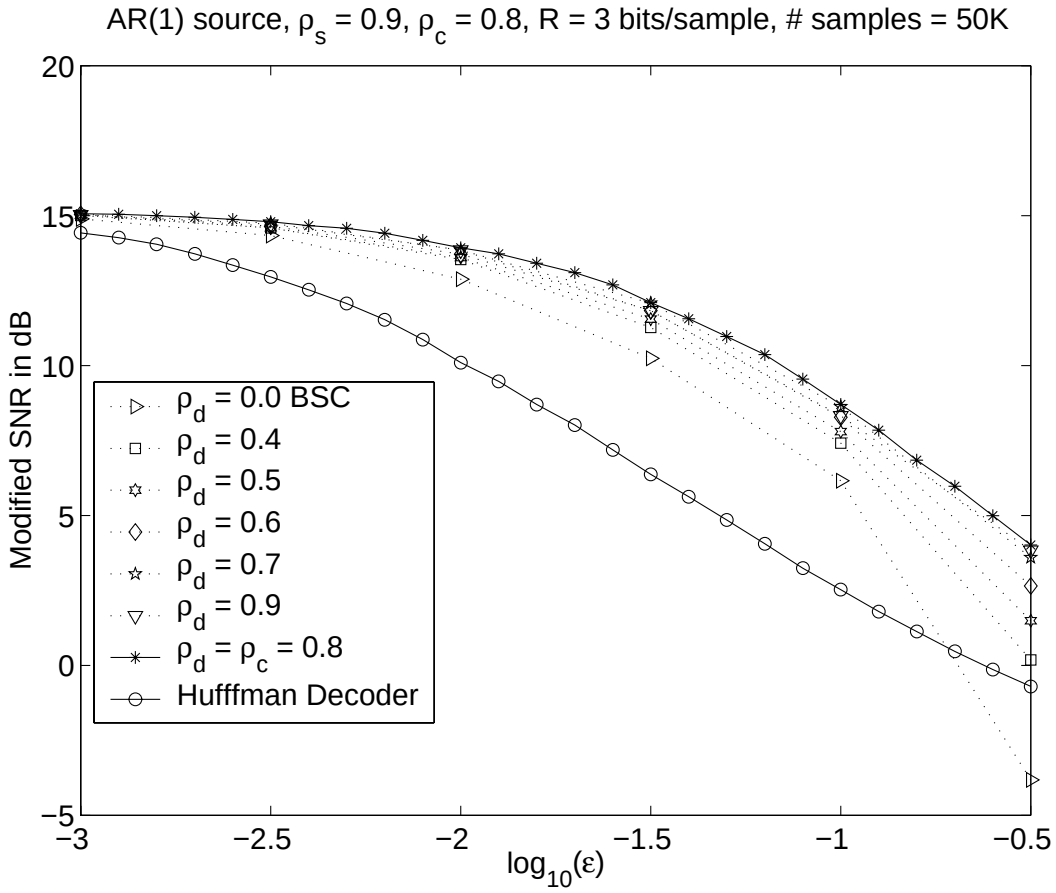


Figure 4.10: MSNR of the MAP-AMC decoder under various channel mismatch conditions compared with the performance of the MAP-AMC under perfectly matched conditions and the performance of the Huffman decoder. $\rho_c = 0.8$ **represents the actual channel correlation and** ρ_d **is the estimated value of the channel correlation coefficient.** $50,000$ **samples of an** $\mathbf{AR}(1)$ **process with correlation coefficient** $\rho_s = 0.9$**, quantized using a** 9 **level uniform quantizer with a step size of** 1.25 **was used.**

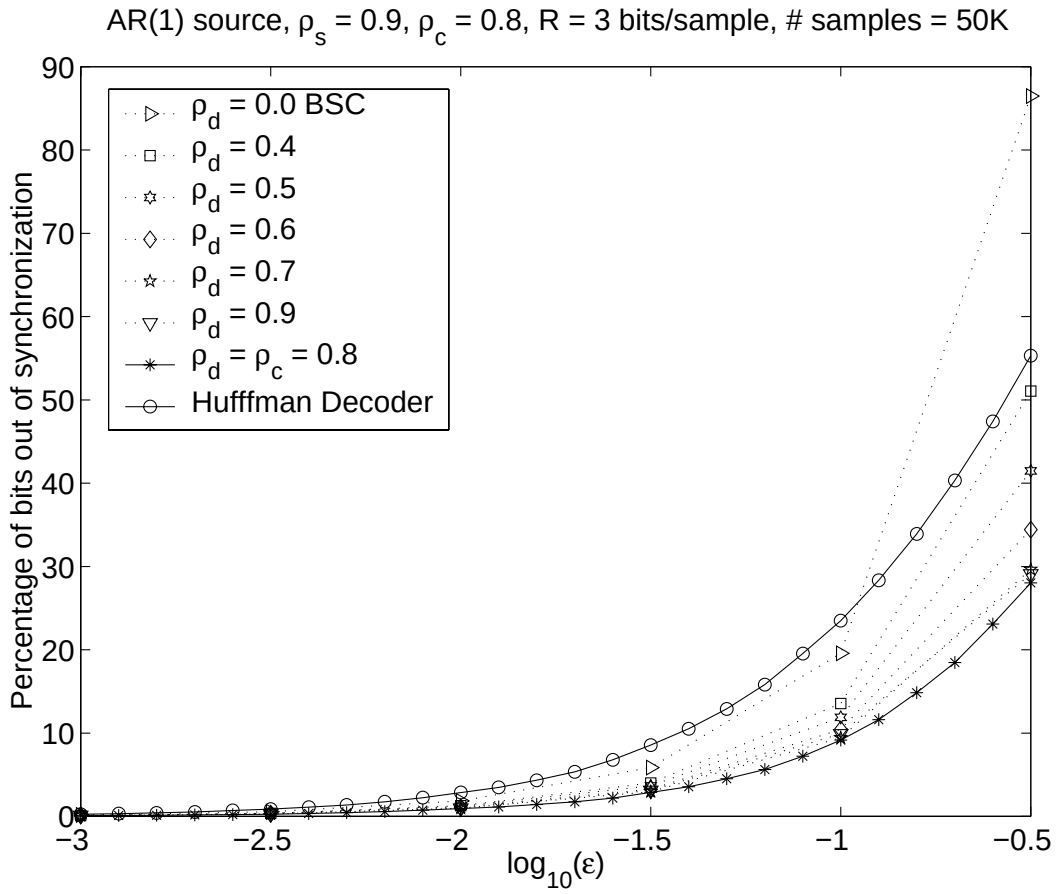


Figure 4.11: Comparison of the percentage loss of synchronization of the MAP-AMC decoded sequence under various channel mismatch conditions with that of the MAP-AMC decoded sequence under perfectly matched conditions and the Huffman decoded sequence. ρ_c represents the actual channel correlation, 0.8 and ρ_d is the estimated value of the channel correlation coefficient. $50,000$ samples of an $\mathbf{AR}(1)$ process with correlation coefficient $\rho_s = 0.9$, quantized using a 9 level uniform quantizer with a step size of 1.25 was used.

4.4.2 MAP-AMC for Memoryless Sources

To study the MAP-AMC decoder performance with memoryless sources, we generated 50,000 samples of a zero-mean, unit-variance Gaussian source and quantized it using a 9-level uniform quantizer having a step-size of 0.55 to produce a rate of 3 bits/sample. The resulting symbol-stream was then entropy-coded and transmitted over six realizations of an AMC with $\rho_c = 0.8$. MSNR and PBOS data were then obtained for a set of channel error-rates. The programs used to implement the codec were essentially the same as in the case of the AR(1) source, except that the training and the transmitting programs generated Gaussian sources and the MAP decoder program implemented the memoryless-source version of the MAP-AMC algorithm. Figures 4.12 and 4.13 are the MSNR and PBOS plots of the MAP-AMC versus the Huffman decoder for this source.

We see that there is neither any redundancy in the form of memory nor that in the form of non-uniformity of pdf that the MAP-AMC is able to make use of and hence we do not find any improvement over the Huffman decoder. As before, we also note that the overall performance is better in comparison to the binary symmetric channel. Our experiments thus indicate that the "believe what you see rule" (the decoder believes that there are no bit errors in the stream and hence parses it as is) performs similar to the optimal MAP solution and that there is no gain in building a MAP decoder for this source. Note that conditions for the optimality of the "believe what you see" rule were derived for the simple case of binary symmetric Markov sources and binary i.i.d sources over additive Markov channels by Alajaji et. al. (1996). Derivation for other, more complicated, sources is a challenging problem and is not attempted in this dissertation.

4.5 Chapter Summary

In summary, this chapter examined the design of a joint source-channel MAP decoder for entropy coded Markov sources transmitted over an AMC. It was seen that the state space developed for the MAP-BSC for Markov sources can be used in the dynamic programming formulation for the MAP-AMC. The MAP-AMC differs from the MAP-BSC mainly in the metric, which involves the channel transition probabilities and

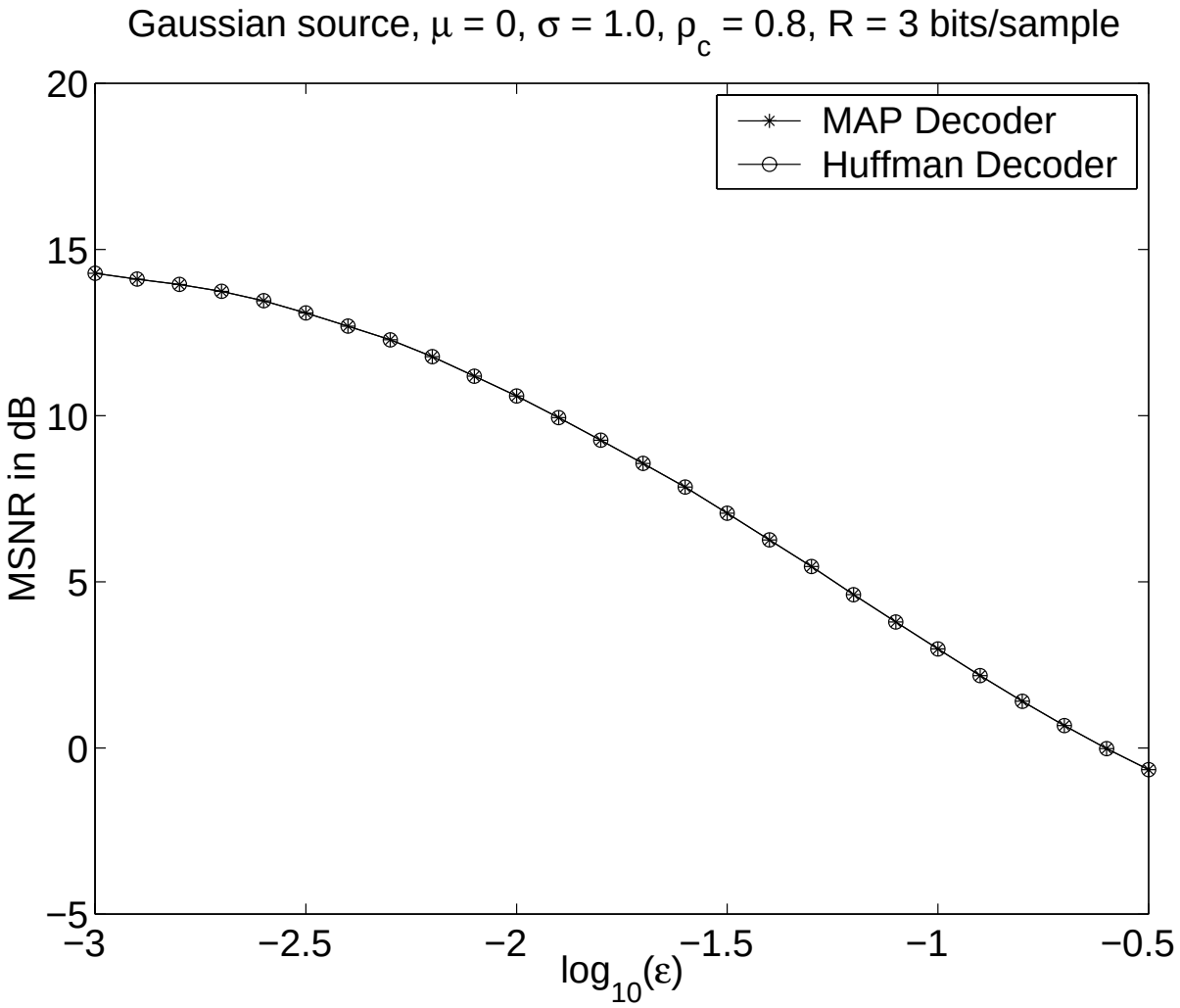


Figure 4.12: MSNR of the MAP-AMC decoder compared to the that of the Huffman decoder for the zero mean, unit variance Gaussian source, quantized at 3 bits per sample with a 9 level uniform quantizer with step size $= 0.55$.

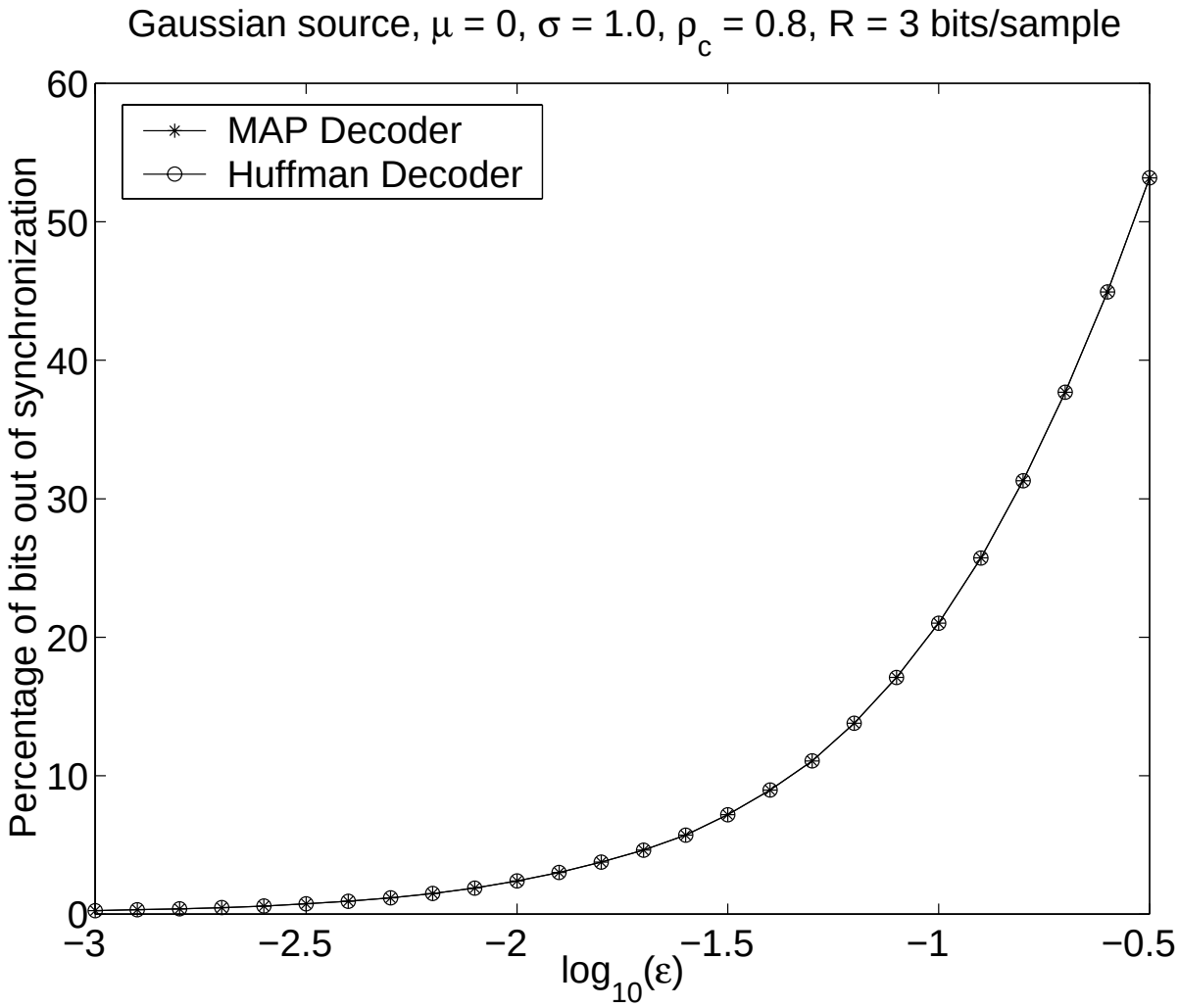


Figure 4.13: PBOS of the MAP-AMC decoder compared to that of the Huffman decoder for the zero mean, unit variance Gaussian source, quantized at 3 bits per sample with a 9 level uniform quantizer with step size = 0.55.

hence requires that the algorithm should remember if the previous bit was erroneous at each stage of the algorithm. The complexity of the algorithm can be considered to be the nearly same as that of the MAP-BSC, since the number of states in the state-space is the same; even though the number of operations may be higher for the MAP-AMC.

Simulation results have demonstrated that this decoder does significantly better than the conventional decoder at all channel error-rates considered. The decoder was also shown to be robust to inaccuracies in the measurement of channel correlation: even when the decoder is designed assuming a BSC-channel ($\rho_d = 0$), we found that the MAP decoder does better than the Huffman decoder at most error-rates in terms of the MSNR. In addition, the overall performance of the MAP-AMC is seen to be significantly better than that of the MAP-BSC.

Chapter 5

Application to Image Transmission

The aim of this chapter is to test the MAP decoders proposed earlier, in a real-life application which uses entropy coding extensively. In particular, the application of the MAP decoders to image transmission is studied in this chapter. The choice of image coding is a bit arbitrary, but low-rate image compression schemes require effective entropy coding to perform well and the statistics of the compressed image data are not well modeled by simple sources and hence images are a more interesting test case.

The effects of synchronization loss can be catastrophic for image coders, since a single pixel dropped or added will cause a displacement of all subsequent pixels which could be very annoying. Although the effective error rate of a channel can be lowered using error control codes, it is important to see how far the performance can be pushed without the use of channel codes, since many practical applications have severe bandwidth constraints to permit the use of these.

To be a good test case for our purposes, a codec will have to employ a variable-length entropy code that can be implemented as a table look up, the statistics of the source must be available and there must be sufficient redundancy in the source in the form of either memory or pdf. We design such a codec in this chapter. The result is not a state-of-the-art image codec (performance is similar to JPEG); however, the structure is sufficient to demonstrate that by combining the MAP-BSC decoder with judiciously placed synchronization symbols, it is possible to obtain good performance, in terms of PSNR, at error rates from 10^{-2} to 10^{-4} without the use of channel codes. Some of the concepts and building blocks used to build this codec were described in Chapter 2.

5.1 Example Codecs

In this section, we describe two "standard" codecs used in image coding and present the reasons why they cannot be applied directly to a JSCD system. The study of these codecs will also motivate the choice of some of the methods that are incorporated in the "custom" codec designed in Section 5.2. Both the codecs in this section belong to the class of *frequency domain* compression. Frequency domain codecs are those that operate on the frequency domain representation of the signal rather than on its time domain representation. In particular, the standard JPEG coder and the SPIHT codec are described in this section.

5.1.1 The JPEG Codec

A joint ISO/CCITT committee group known as JPEG (Joint Photographic Experts Group) was set up in the mid 1980s to study an efficient coding scheme for continuous-tone still images. This body developed an image compression scheme widely known as the JPEG standard for still image compression.

The JPEG scheme (Pennebaker and Mitchell 1993) uses the discrete cosine transform (DCT) to decompose the image signal into various "frequency" components. Typically (for natural images), the DCT concentrates most of the energy of the image, into its DC component.

What follows is a brief description of the baseline JPEG codec, which is one of the basic modes of operation for the codec. The image is first level-shifted so that the neutral gray intensity is changed to zero. The level-shifted image is then partitioned into 8×8 blocks in a raster scan fashion. A 2D-DCT is computed for each 8×8 block resulting in one DC component and 63 AC coefficients per block. Since the DC coefficients are statistically different from the AC coefficients, they are treated separately. In particular, the DC coefficients are differentially coded by the following first order prediction

$$\Delta = \mathrm{DC}_i - \mathrm{DC}_{(i-1)} \ , \tag{5.1}$$

where DC_i is the DC coefficient of the i^{th} image block. The result is then scalar quantized using a scalar quantizer specified by a table. The AC coefficients are scalar quantized using another quantization table. The quantized coefficients are scanned in

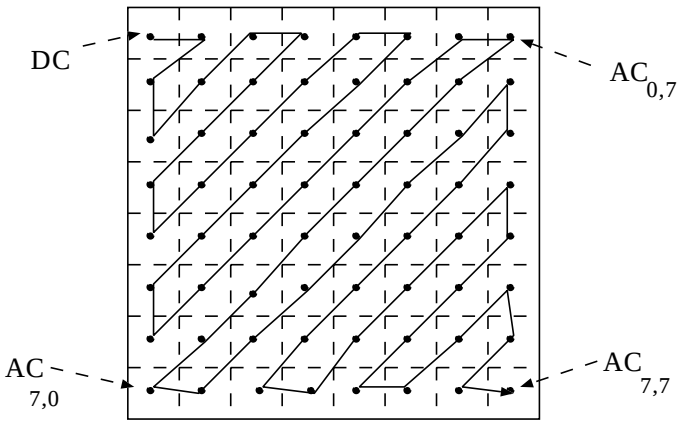


Figure 5.1: Zig-zag scan of quantized DCT coefficients for one 8×8 block in an image

a zig-zag fashion, as shown in Figure 5.1, which orders the coefficients according to increasing spatial frequency and, in general, decreasing variance. An efficient variable length code table that represents runs of zero coefficients that is followed by the size of the non-zero coefficient, is developed as part of the standard. JPEG uses a Huffman encoder for its baseline system, with maximum codeword length of 16 bits. In general the JPEG image coder performs reasonably well for higher rates, the preferred compression techniques for lower rates (< 0.5bpp) tend to be based on subband decompositions, which provide better PSNRs and fewer annoying artifacts (such as blocking). In fact, the latest JPEG image coding standard, JPEG 2000 (based on the EBCOT algorithm (Taubman)), is a subband based coder. Good sources for further information on the JPEG codec are (Pennebaker and Mitchell 1993; Gibson, Berger, Lookabaugh, Lindebergh, and Baker 1998) and (Rao and Hwang 1996).

The sequence of AC coefficients is mapped into an intermediate sequence of what are called "symbol-1" and "symbol-2" pairs. Symbol-1 consists of (RUNLENGTH, SIZE), where RUNLENGTH is the length of zero values preceding the next nonzero AC coefficient amplitude, which is symbol-2. So, symbol-2 is (AMPLITUDE), which is the value of the AC coefficient. The symbol-1 sequence is entropy coded, Huffman

coded in the baseline version, but symbol-2 is assigned its direct binary representation if positive, or the one's complement representation if it is negative (Gibson, Berger, Lookabaugh, Lindebergh, and Baker 1998).

Even though JPEG involves a variable length encoded bit-stream, it is not the symbols that are directly encoded. In the JPEG codec, the spatial memory of the image is broken up; first by 8×8 blocks and then by implementing a run-length code. This could substantially reduce the redundancy in the encoded source implying that there may not be much improvement on using the MAP-BSC decoder in this scenario (cf Section 3.5.2). However, the study of the above mentioned redundancy and the applicability of the JSCD decoder to the JPEG codec is left for future work.

5.1.2 The SPIHT Codec

The SPIHT algorithm (Said and Pearlman 1996) is a popular wavelet based image coding scheme based on set partitioning in hierarchical trees. SPIHT uses octave-band filter banks in the subband decomposition of the image and the variance of the coefficients decreases from the highest to the lowest bands in the subband pyramid. This scheme is an improvement of the embedded zero tree coding algorithm or the EZW algorithm due to Shapiro (1993). Both schemes produce an embedded bit stream of data, which means that the decoding can be stopped at any point and the image can be decoded to some level of accuracy. The difference between the SPIHT and EZW algorithms, is that the SPIHT algorithm provides its top performance, for all rates, with one embedded bit-stream, while the EZW algorithm needs some parameters to be optimized for each rate. Also, the EZW algorithm requires arithmetic encoding of the bit stream for good compression, whereas the SPIHT algorithm can perform reasonably well even without arithmetic coding. This is so because the subset partitioning is very effective in the SPIHT algorithm and the resulting significance information is very compact.

The crux of the SPIHT algorithm lies in ordering the subband coefficients according to their magnitude and transmitting the most significant bits of the significant coefficients first. The algorithm is based on three concepts: 1) partial ordering of the transformed image coefficients by magnitude, with transmission of order by a subset partitioning algorithm that is duplicated at the decoder end, 2) ordered bit plane

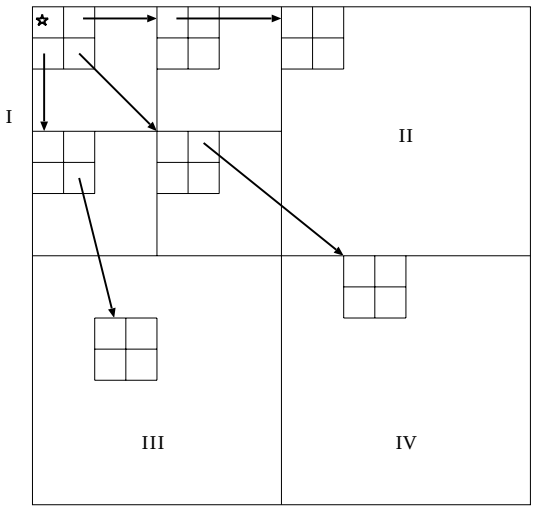


Figure 5.2: Examples of parent-offspring dependencies in the spatial-orientation tree.

transmission of refinement bits, and 3) exploitation of self-similarity of the wavelet coefficients across different scales. The wavelet coefficients are expected to be better magnitude-ordered if we move down 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 at the lower levels at the same spatial locations. An example of the dependencies of the coefficients across different scales is shown in Figure 5.2. In the figure, the outermost rectangle represents the 2-D frequency plane of the image. The first filtering operation (in the octave-band splitting, cf. Section 2.5.1) produces the first four (largest) quadrants in the figure. Since an image is a 2-D signal, one might use different filters for the horizontal and the vertical directions. If we classify the filters into the low-pass and the high-pass filters, then we have four possible frequency bands corresponding to the four possible combinations of the filters and the directions. Hence, the LL-band (I quadrant in the Figure 5.2) corresponds to filtering the image in both directions using a low-pass filter, the II quadrant corresponds to the low-high band and so on.

The algorithm can be basically divided into the *sorting pass* and the *refinement*

pass. The sorting pass uses what is known as the *spatial orientation tree*. This tree is built based on the observation that there is a spatial self-similarity between the subbands (see Fig 5.2). Each node in the tree corresponds to a pixel and is identified by the pixel coordinate. Its direct descendants correspond to pixels of the same spatial orientation in the next finer level of the pyramid. In Fig 5.2, the arrows are oriented from the parent to its four offsprings. Parts of the spatial orientation tree are used as the partitioning subsets in the sorting algorithm. The sorting pass orders the coefficients according to their significance, which is judged by the amount of reduction in distortion that can be gained upon receiving a coefficient at the decoder. The refinement pass transmits the most significant bit of all the coefficients contained in the list of significant pixels, except those included in the previous sorting pass. A large fraction of the "bit-budget" is spent on the sorting pass and it is there that the sophisticated coding methods are needed, hence the quantizer can be kept simple (e.g a scalar quantizer).

One of the main features of the algorithm is that the ordering data is not explicitly transmitted. Instead, this information is inferred from the fact that the execution path of any algorithm is defined by the results of the comparisons on its branching points.

If arithmetic coding is not used in the SPIHT algorithm, then the decoder presented in this thesis is not applicable to the situation, since there is no variable length encoded sequence involved in the codec. On the other hand, if arithmetic codes are, indeed used to encode the bit streams, then we do use a variable length encoder in the codec, but as discussed in Section 2.5.3, it may not be very useful to design a MAP decoder for a codec that uses arithmetic codes.

5.2 Codec Design

This section deals with the design of the image codec used in evaluating the MAP-BSC decoder of Chapter 3. As pointed out in previous sections of this chapter, the MAP-BSC decoder requires the existence of residual redundancy in the form of source memory or a peaky pdf and the entropy coder used must be implementable in a table look-up fashion. In what follows, we will discuss the design of such an image codec and look at the issues that dictated the parameters that we chose.

The image is decomposed using a wavelet transform, quantized, Huffman encoded

and transmitted. At the receiver end a MAP-BSC decoder is implemented and the decoded image is reconstructed. The following subsections deal with each aspect of the codec, starting with the wavelet transform that was used.

5.2.1 The Wavelet Transform

Antonini et al. (1992) developed a biorthogonal wavelet based image compression scheme using three sets of filter banks: a set of filters where the low-pass and high-pass filters have dissimilar lengths (9-3), a set of filters where the low and high pass filters have similar lengths (9-7) and a set of filters where the analysis and synthesis wavelets are similar (5-7). It was shown that the performance possible with the the 9-7 filters was better than the current state-of-the-art codecs. In fact, the 9-7 filter set is still widely used, has been incorporated in the JPEG 2000 standard (Taubman) and is considered one of the best overall filter sets for image compression. In order to demonstrate the effective application of our MAP-BSC algorithm to images, we designed a simple image codec based on a wavelet-transform using the above mentioned "standard" 9-7 filter set (tap coefficients are given in Table 5.1). The Antonini codec

Filters	0	±1	±2	±3	±4
$h_0(n)$ $(\times 10^{-2})$	8.53	3.77	-1.11	-2.39	3.78
$g_0(n)$ $(\times 10^{-2})$	7.89	4.18	-4.07	-6.45	0

Table 5.1: Taps of the Antonini 9-7 filters

uses a multiresolution VQ codebook for each of the bands in the decomposition. This means that for a 3-level decomposition, which produces 10 subbands, we would need to send 10 different sets of probabilities (conditional and marginal) to the decoder as overhead. We design a codec that reduces this overhead, as will be seen in the next few subsections.

We note that even though we use a 3-level decomposition scheme in this chapter, it is possible to increase the number of levels in the decomposition; however, a couple of issues will have to be kept in mind while doing this. One of the issues is the number of different Huffman codebooks and the associated marginal and transition probabilities that need to be transmitted to the decoder as side information. As will

be described in greater detail in Subsection 5.2.2, each level in the codec is entropy coded using a different Huffman codebook and, in order for the decoder to function, these codebooks must be transmitted as side information. Greater decomposition depths thus require more side information. Also, the number of data points in each sub-image decreases with each increase in the decomposition depth (as seen in Fig 5.3) and hence probability estimates become less accurate. In the experiments with our codec structure, it was found that increasing the number of levels from three to four only increased the performance of the system by about 0.2 dB. Hence, we used the less complex 3-level system, which makes it easier to focus on the issues that are unique to JSCD. Enhancements to the performance of the codec are left for future work.

5.2.2 Quantization and Bit Allocation

Vaisey et al. (1998) showed that good rate-distortion performance is possible simply by quantizing each subband with its own uniform scalar-quantizer followed by arithmetic coding. In fact, for the "Lenna" image at 0.4bpp, their algorithm performs just 0.7dB below the SPIHT algorithm. The trick is to find appropriate step sizes using a bit-allocation procedure like the generalized BFOS algorithm. However, as we discussed in Section 2.5.3, implementing arithmetic codes in a table-look up fashion may not be very effective for our purposes and hence we do not use this method in our codec. Moreover, if Huffman codes were applied to each coefficient in each of the subbands, the resultant bit rate cannot go below one bit for each coefficient because the smallest codebook is the one bit codebook. This rate is much too high and hence we group some of the coefficients into vectors and design a VQ which will allow us to achieve fractional rates.

In our codec, the low frequency, LL-band, is encoded using a uniform quantizer, but the high-pass bands (HP-bands) are encoded using 3D vector quantizers. The LL-band us treated separately because of its very different characteristics. We choose to scalar quantize the LL-band because the number of samples in the LL-band is too small to construct large vectors and a scalar quantizer is a viable alternative to a VQ.

Each vector of HP-band coefficients is formed by grouping the three high-pass subband coefficients having the same resolution and spatial position, as shown in Fig 5.3. The reason for such a grouping is that the three coefficients that are grouped

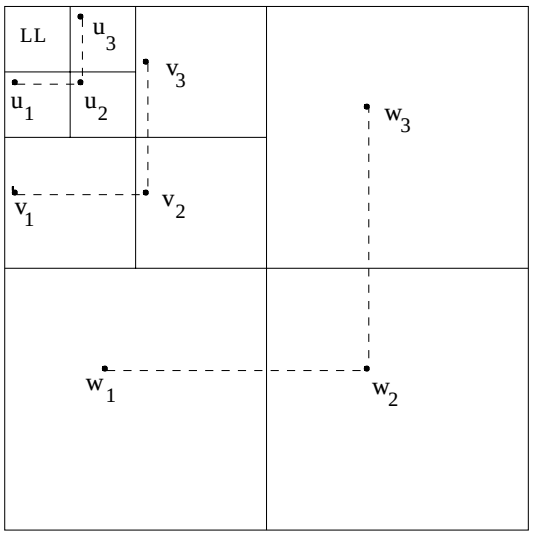


Figure 5.3: Vector formation for higher frequency bands

together bear a relationship to each other as can be seen from Fig 5.4 which shows the 3-level subband decomposition of the "Barb" image. Also, these three bands have similarly shaped probability distributions that are quite different from the LL-band, as will be seen in Section 5.2.6 when discussing the residual redundancy in the source. Lastly, this strategy also keeps the memory between adjacent vectors accessible to the MAP decoder since they are still direct spatial neighbours. It is possible to construct a vector of dimension larger than three by taking more than one sample in a single band. This could potentially allow for higher compression but larger vector sizes would also imply smoothing in the higher bands. For a three-level subband decomposition, this grouping produces four sub-sources that need to be transmitted. For notational convenience, we number these sub-sources from 0 through 3, with the LL-band being the sub-source 0, the sequence of vectors (like $\mathbf{u} = [u_1, u_2, u_3]$ in Fig 5.3) from the third level forming the sub-source 1 and so on.

Entropy constrained vector quantizers (ECVQs) (cf. Section 2.5.2) were developed for the vector sources with 64 codewords in each codebook and the LL-band itself was scalar quantized with 64 level uniform quantizers. Note that increasing the number

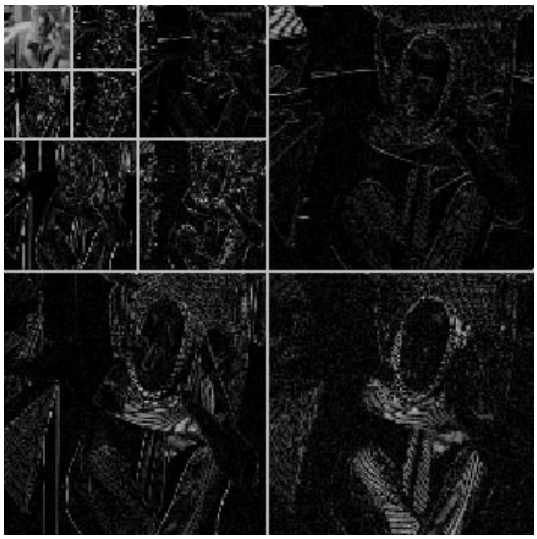


Figure 5.4: 3-level decomposition of the "Barb" image.

of codebooks in the ECVQ codebook does not automatically imply an increase in the number of bits when entropy coding is used. We chose to use 64 codewords as they gave better R-D performance than when we used fewer number of codewords and increasing the number of codewords did not give rise to substantially better R-D performance for the rates we were interested in (between 0.5 and 0.6bpp). Also, increasing the number of codewords will imply an increase in the side information that needs to be sent to the decoder (cf. Section 5.2.4).

Two test images, "Lenna" (Fig. 5.5) and "Barb"(Fig.5.6), were considered in the experiments. In choosing test images it is usual to look for images that are characteristic of a large class of images. "Lenna" does not have a lot of energy in the high-pass bands and is therefore easier to code. "Barb", on the other hand, has significant energy in the HP-bands and is much more difficult to code. We picked these as test images, so as to make the tests more generic and because these are also commonly used as test images in the image coding literature.

As discussed briefly in Section 2.5.2, the choice of training images is important in

Figure 5.5: Test Image: "Lenna"

Figure 5.6: Test Image: "Barb"

Baboon Clown Mire

Figure 5.7: Training Images for "Barb"

the design of VQs. Since, VQ is essentially a pattern matching algorithm, the codebook must contain enough patterns that are close enough to the image that needs to be encoded. Hence, the tendency is to pick training images that are representative of the images that are to be encoded. For example, the "Barb" image has high frequency components in it, which are in general more difficult to code. So, the training images picked for constructing the ECVQ for the "Barb" image were: "baboon", "clown" and "mire" (shown in Fig. 5.7), each of which also display some high frequency components. The training images for ECVQs for the "Lenna" image (which has more low frequency components than high frequency components) were "grandma", "clown" and "man" (shown in Fig. 5.8) which taken together, have a smaller concentration of high frequency components.

Bit allocation between the four sub-sources was done by designing families of VQ codebooks for each vector quantizer, a set of scalar quantizers for the LL-band and their corresponding Huffman codes (to get the values for the rates) and then applying the generalized BFOS algorithm (Riskin 1991) to produce a system operating at close to the target rate. The rate-distortion table used in the generalized BFOS algorithm for each of these sub-sources for the "Lenna" image is given in Table 5.2 and that for the "Barb" image is given in Table 5.3. The corresponding operational R-D curves for sub-sources 0 through 3 are plotted in Figures 5.9 through 5.12, for the "Lenna"

Grandma Clown Man

Figure 5.8: Training Images for "Lenna"

Sub-source 0			**Sub-source 1**			**Sub-source 2**			**Sub-source 3**		
Dist.	Rate	Δ	Dist.	Rate	λ	Dist.	Rate	λ	Dist.	Rate	λ
642.494	4.037	90	244.843	1.129	150	63.124	0.543	150	20.364	0.338	150
534.900	4.198	80	241.011	1.246	120	56.374	0.584	120	19.275	0.340	120
404.670	4.367	70	238.680	1.285	100	51.689	0.617	100	18.187	0.343	100
306.541	4.571	60	234.928	1.385	50	40.260	0.758	50	14.564	0.365	50
207.229	4.822	50	234.565	1.398	40	38.113	0.811	40	13.148	0.382	40
134.821	5.136	40	233.627	1.431	20	32.815	1.132	20	9.848	0.456	20
75.391	5.498	30	233.545	1.446	10	31.426	1.304	10	6.989	0.626	10
60.898	5.643	27				30.938	1.435	1	3.086	1.557	1
51.206	5.745	25									
49.449	5.797	26									

Table 5.2: The rate and distortion values for the families of quantizers designed for the 4 sub-sources of the "Lena" image

Sub-source 0			**Sub-source 1**			**Sub-source 2**			**Sub-source 3**		
Dist.	Rate	Δ	Dist.	Rate	λ	Dist.	Rate	λ	Dist.	Rate	λ
190.78	5.05	48	674.50	0.585	1500	237.09	0.52	500	64.05	0.42	200
179.62	5.10	46	604.81	0.627	1250	167.71	0.65	300	57.95	0.44	170
155.22	5.16	44	556.00	0.661	1125	160.08	0.67	270	54.08	0.45	150
149.37	5.22	42	518.37	0.691	1000	157.78	0.68	260	51.44	0.46	135
136.06	5.29	40	483.23	0.724	825	155.13	0.69	250	49.77	0.47	125
119.62	5.35	38	447.41	0.762	750	153.17	0.70	240	48.26	0.48	115
106.63	5.42	36	405.42	0.827	625	151.56	0.70	230	45.55	0.49	105
96.72	5.50	34	327.73	1.070	500	149.96	0.71	220	44.03	0.50	95
83.17	5.59	32	321.74	1.094	450	147.76	0.72	210	42.61	0.51	85
82.65	5.63	31	316.48	1.115	420	146.96	0.73	205	41.42	0.53	75
76.23	5.67	30	313.23	1.129	400	145.39	0.73	200	40.38	0.54	65
69.72	5.72	29	311.03	1.139	390	139.07	0.77	170	39.70	0.55	60
64.36	5.77	28	309.80	1.146	380	135.13	0.79	150	39.16	0.56	55
59.60	5.81	27	307.43	1.159	360	126.50	0.88	105	38.86	0.57	50
57.30	5.86	26	304.90	1.172	340	122.73	0.93	75	36.82	0.64	25
			303.45	1.180	330	120.15	0.99	50	35.85	0.74	12.5
			302.08	1.187	320	117.50	1.10	12.5	35.48	0.80	6.25
			299.38	1.203	300	117.30	1.13	6.0	35.32	0.86	1
			297.68	1.214	280	117.27	1.134	3			
			297.11	1.219	270	117.25	1.141	1			
			296.67	1.224	260						
			295.45	1.233	250						
			294.54	1.241	240						
			293.98	1.246	230						
			293.46	1.252	220						
			292.56	1.261	210						
			292.22	1.264	205						
			291.89	1.266	200						
			289.93	1.279	170						
			288.98	1.286	150						
			286.96	1.308	105						
			285.71	1.326	75						
			285.00	1.339	50						
			284.66	1.350	25						
			284.51	1.358	12.5						
			284.44	1.361	6.25						
			284.33	1.364	1						

Table 5.3: The rate and distortion values for the families of quantizers designed for the 4 sub-sources of the "Barb" image

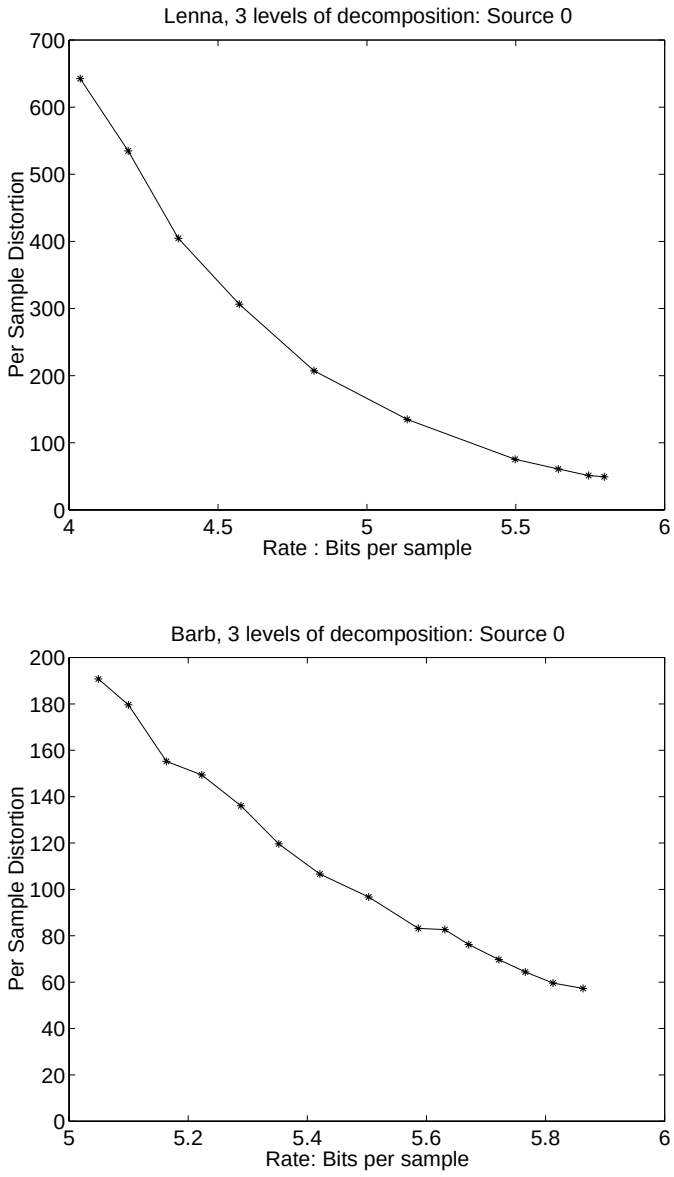


Figure 5.9: R-D Plot for the sub-source 0 for the "Lenna" and "Barb" images

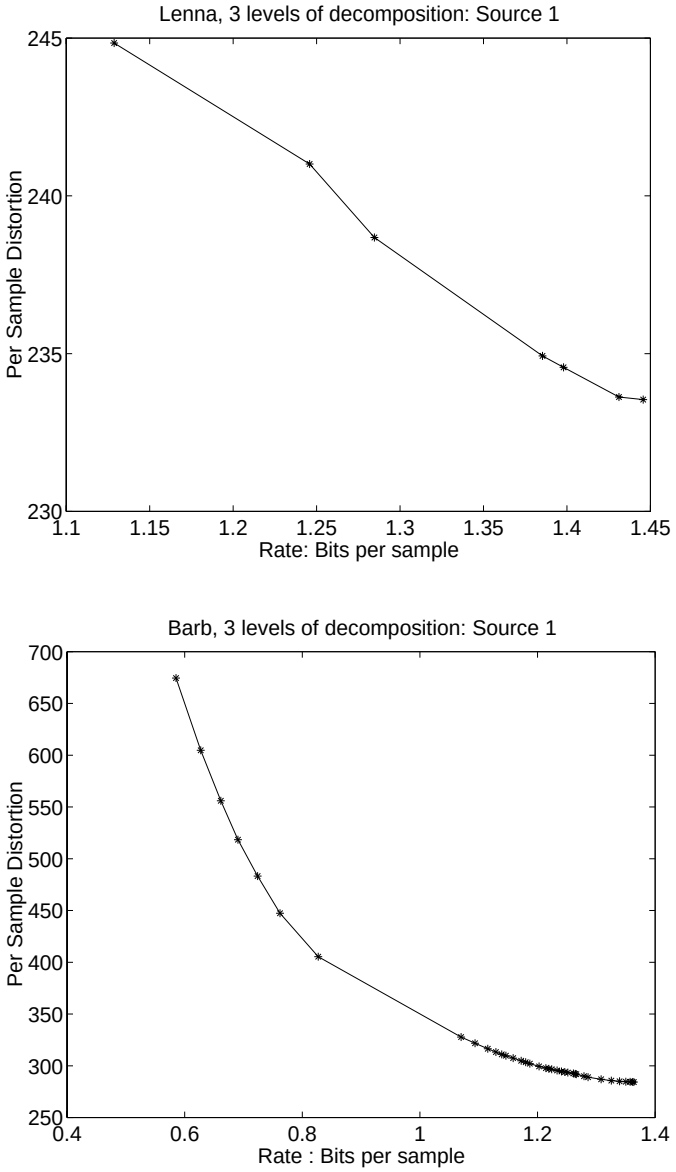


Figure 5.10: R-D Plot for the sub-source 1 for the "Lenna" and "Barb" images

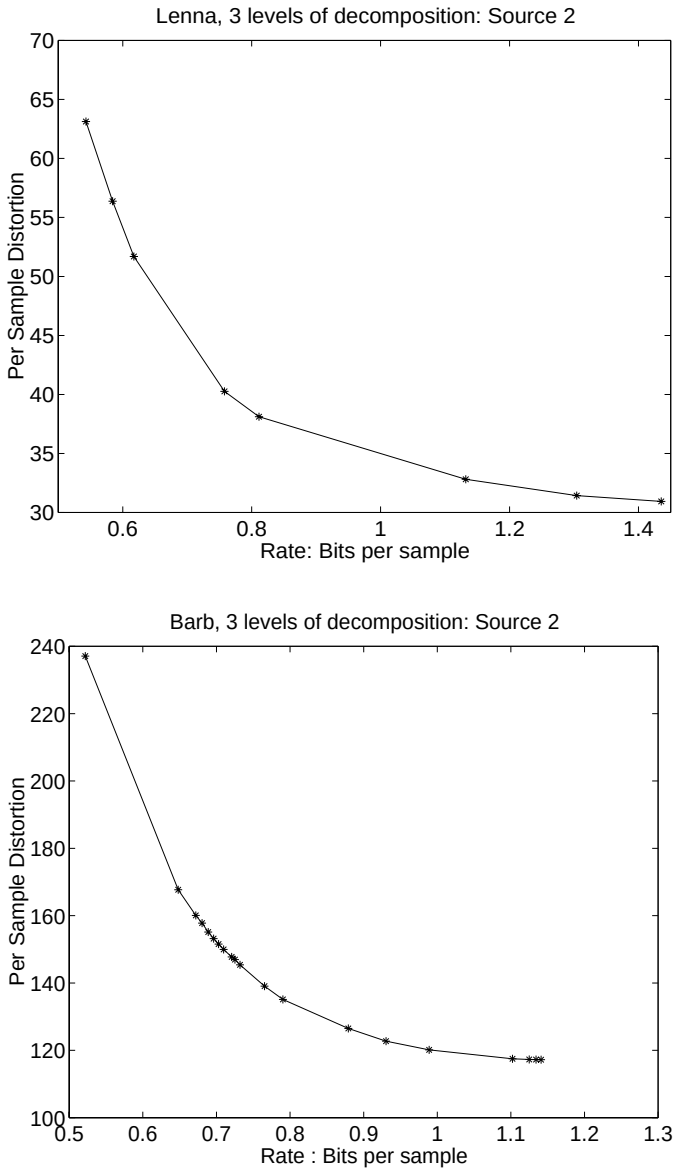


Figure 5.11: R-D Plot for the sub-source 2 for the "Lenna" and "Barb" images

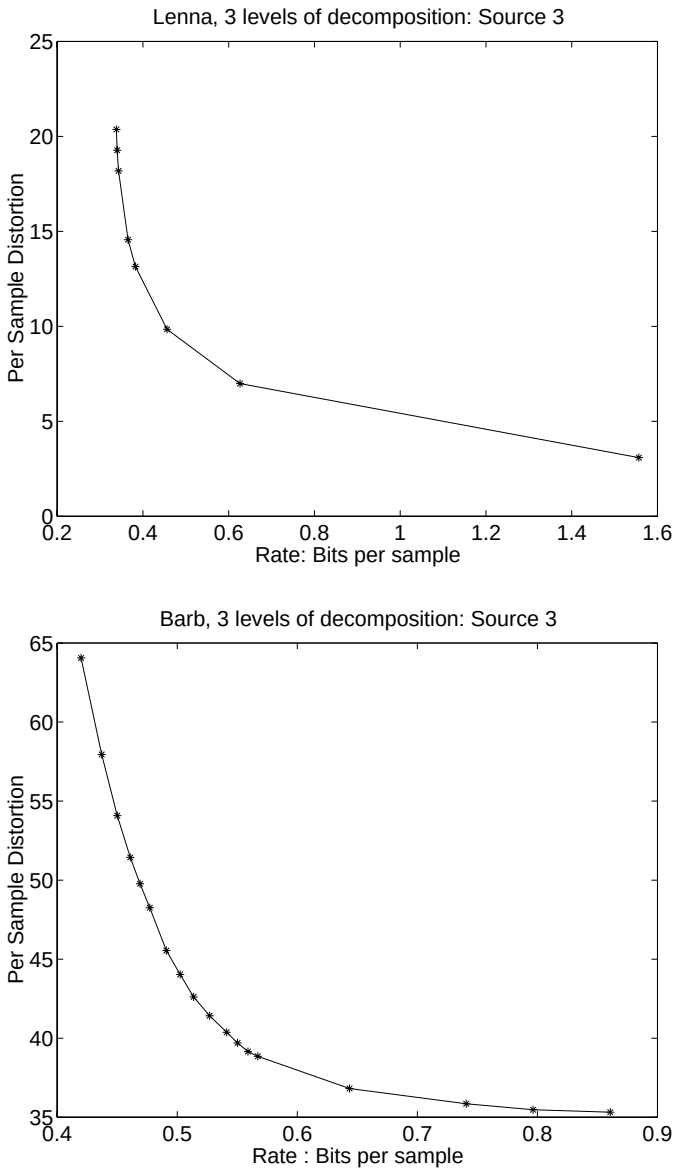


Figure 5.12: R-D Plot for the sub-source 3 for the "Lenna" and "Barb" images

and "Barb" images. Please note that the vertical and horizontal scales are not equal in all images. The tables also list the values of the step-sizes (Δ) for the scalar quantizers and the values of the parameter λ (see Section 2.5.2) for the ECVQs for which these rates and distortions were obtained. We emphasize again that the rates used in the BFOS algorithm are the true rates and not the entropy of the source. The BFOS algorithm lists the quantizers to be used for each source for the requested rate. If the quantizer turns out to be at either ends of the table, then there is a possibility that the BFOS is forced to pick this quantizer as there is no other quantizer with a higher (or lower) rate in the table. This was used to decide how many entries to have in the table: if the BFOS algorithm picked the entry at either end of the table, we went back and designed a few more quantizers.

While using the generalized BFOS algorithm it must be noted that each of the sub-sources has a different number of symbols in them and hence, the distortions and rates have to weighted appropriately when making pruning decisions.

5.2.3 EOL Symbols

End-of-line (EOL) symbols are used to limit error propagation through variable-length encoded signals. As mentioned in Chapter 2, in this approach, a special symbol (often highly protected using forward error correcting codes) is transmitted periodically. The correct receipt of this symbol lets the decoder regain synchronization with the encoder, since the bit pattern is unique and no other codeword is allowed to be contained in the special codeword. EOL symbols are often used in image transmission schemes, since images are much more sensitive to loss of synchronization than signals like speech, which are oriented temporally rather than spatially.

Designing EOL symbols is an interesting problem in itself. Most EOL symbol design techniques published in the literature have been arbitrary and designed for the specific variable-length codes used. Such a design procedure makes the EOL symbols extremely long. Lei et al. (1991) propose an algorithm that has some control over the length of the EOL codewords it designs. Since the design of EOLs can be a reasonably challenging problem in itself, we do not address it in this dissertation. In fact, we do not design an explicit EOL symbol for our purposes, we merely assume that the EOL codeword is well protected and that it is always decoded correctly.

In our codec scheme, for the LL-band, we assume that an EOL symbol is transmitted at the end of every line. For the higher frequency bands, we assume that an EOL symbol is transmitted at the end of every line of *vectors*. Note, that it is possible to change the number of EOL symbols in the scheme. Decreasing the frequency of EOL symbols increases the overhead, but also increases the sensitivity to errors.

5.2.4 Implementing the Codec

Figures 5.13 and 5.14 depict the encoder and decoder that is used in our experiments. As shown in the figure, the image codec is wavelet based. In the Fig 5.13, the solid lines correspond to the actual data and the various broken lines represent side information. As can be seen from the diagram, the image is first subband decomposed to three levels and four sub-sources are formed as detailed in Section 5.2.2.

The training phase, where three vector quantizers $\mathrm{VQ}_1, \mathrm{VQ}_2$ and VQ_3 were designed for the three vector sub-sources, was implemented as a separate program which also designs the Huffman codebooks for all the sub-sources based on the actual image to be transmitted and writes out all these in separate files. This program also encodes the image to be transmitted using the codebooks designed and writes out the Huffman encoded bit streams in separate files (one for each sub-source). A separate file is generated which records the length (in bits) of each line in the image sub-source. As pointed out in the previous section, we did not implement an EOL symbol, we assume that it is always decoded correctly and hence we only need to know where each line in the sub-band ends, which is known if the length of each line is known. The statistics (marginal and conditional probabilities) of the sub-sources are calculated for the image to be transmitted and written out as separate files. Note that both the Huffman codes and the statistics may be calculated from the training set instead of for each image that needs to be transmitted, in which case only a one time transmission of these parameters is required as opposed to transmitting one set of values for each image to be transmitted. On the flip side, the statistics will be less accurate and this may affect the performance of the MAP decoder.

The BSC was implemented as a separate program to which the Huffman encoded sub-sources were input and which outputs the bit streams corrupted according to the channel error probability ϵ.

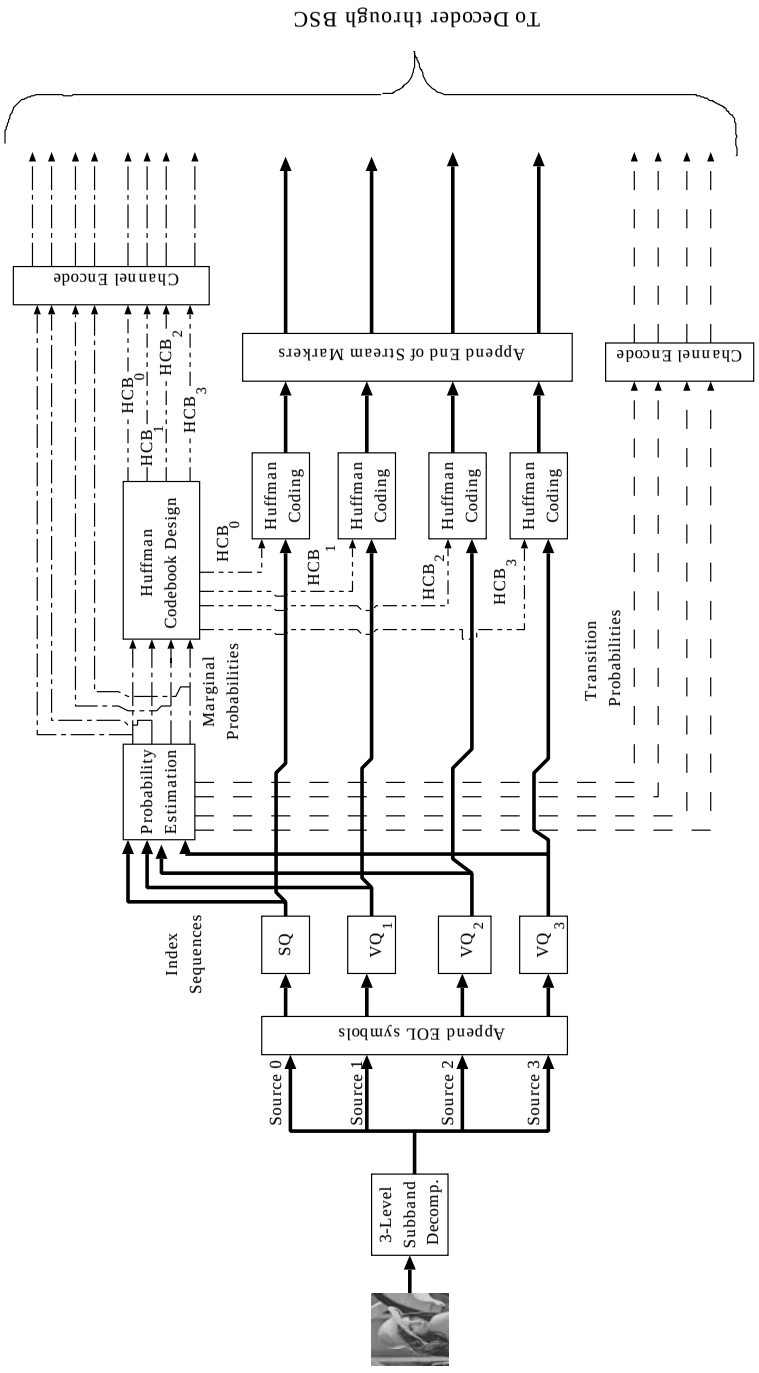


Figure 5.13: The Encoder Used for the Experiments on Images

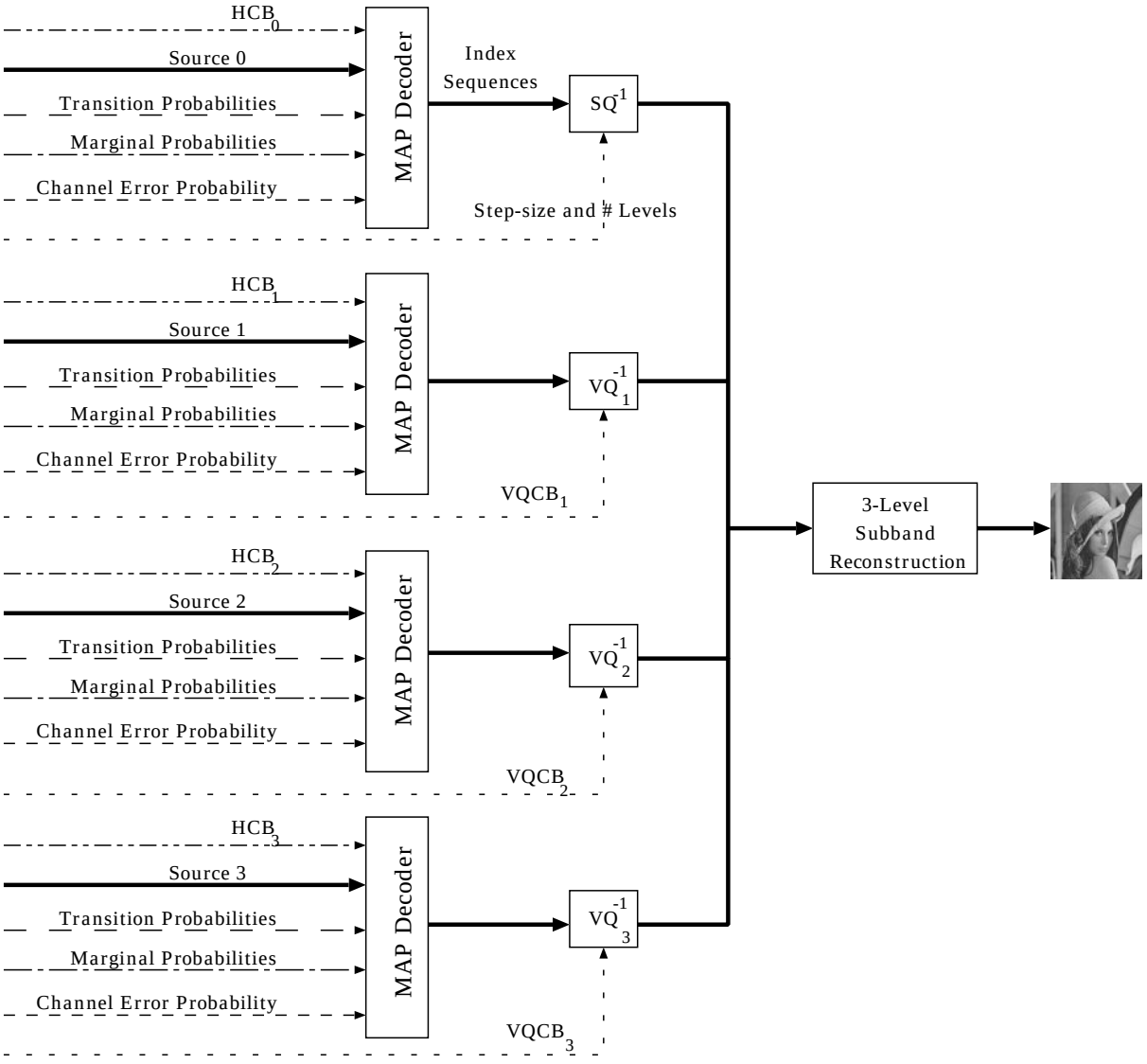


Figure 5.14: The Decoder Used for the Experiments on Images

The decoder reads in the files corresponding to all the side information and the corrupted bit stream and implements the MAP decoder for each line in the sub-sources. Note that ϵ is also input to the decoder. Again, we assume that the side information has been sufficiently protected so that it can be correctly decoded at the receiver. We do not explicitly implement any channel code. Note that the extra side-information that is required by the JSCD decoder, over and above any non-JSCD decoder are the the marginal and conditional probabilities of each of the sub-sources and the channel error probability ϵ.

5.2.5 Performance of the Codec

Figures 5.15 and 5.16 compare the performance of the codec designed in this chapter with the JPEG and SPIHT codecs for the "Lenna" and the "Barb" images respectively. The codec was tested by implementing the encoder and the decoder as separate programs, with the encoder's output being fed to the decoder in separate files.

As can be seen from the graphs, the SPIHT codec does better than the other two codecs for both the images at all error rates considered. This result is expected as SPIHT is a much more sophisticated algorithm than the other two. On the other hand, it can also be seen that the baseline JPEG does better than our codec in compressing the "Lenna" image; whereas, it is either worse than or marginally better than our codec for the "Barb" image.

The difference in performance of the codec for the two images can be explained as follows. The lowest rate at which any of the vector sources can be transmitted is 1/3 in our case. This is because there has to be at least one bit to represent a vector and there are three source symbols to a vector. Sometimes, transmitting these vectors may not be "worth" the increase in bit rate. For example, the energy in sub-source 3 for the "Lenna" image was found to be 25.85 per sample; whereas for the "Barb" image it was 200.36 per sample. So, if the sub-source 3 were not transmitted for the "Lenna" image the distortion goes up by only 25.85 per sample. This penalty may be worth paying for a good R-D performance. In fact, new R-D tables were constructed for the "Lenna" image with the option of not transmitting the sub-source 3. Bit allocation was performed using this table and it was seen that the performance of our codec was comparable to the JPEG codec at a rate of 0.43 bpp; when our codec gave a PSNR of

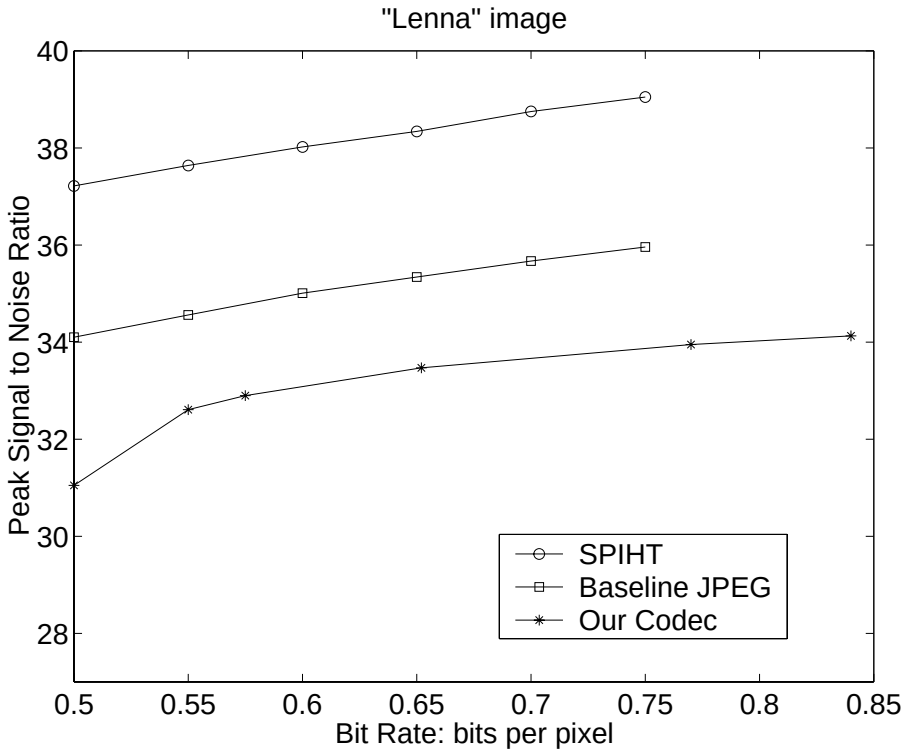


Figure 5.15: Comparison of the different codecs for the "Lenna" image

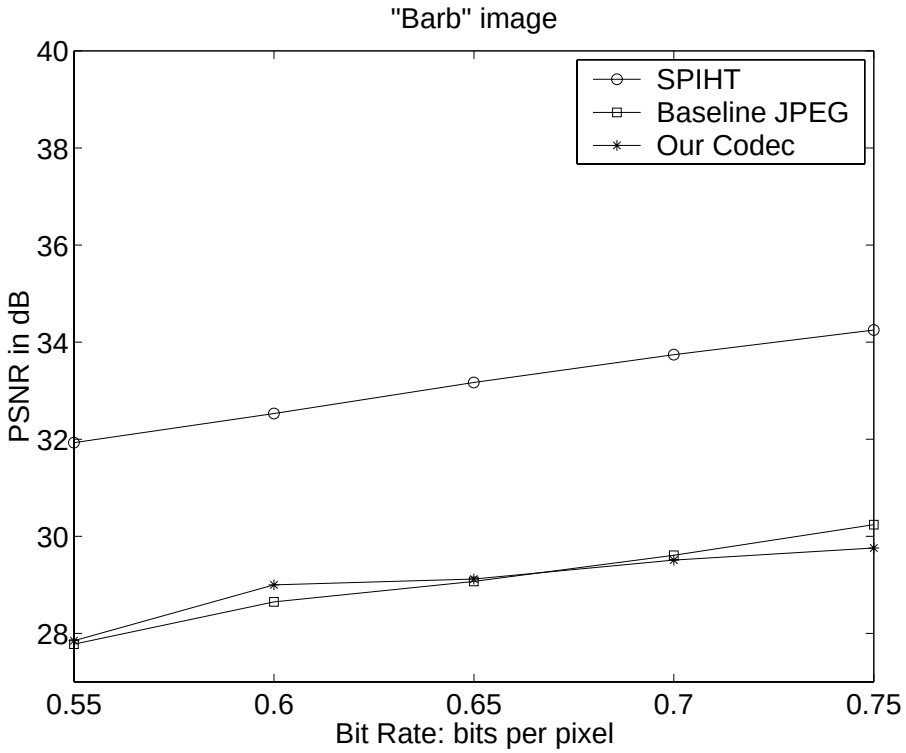


Figure 5.16: Comparison of the different codecs for the "Barb" image

33.31 dB and the JPEG codec gave a PSNR of 33.22 dB for the "Lenna" image. So, it may be safe to conclude that our codec can perform similar to the JPEG codec if there is a provision to not transmit some of the lower energy sub-sources. We would like to reiterate that the object of the chapter is not to design a state-of-the-art codec but to test our 1-D JSCD algorithm in a more practical situation.

The following subsection deals with the way in which this 1-D JSCD algorithm can be adapted to 2-D images and discusses the reasons why a performance improvement can be expected from this method.

5.2.6 The Decoder

As seen from the Figures 5.13 and 5.14, the four output streams of the encoder are passed through a BSC and are decoded by a MAP decoder at the receiver end. The JSCD algorithm models the streams of quantization indices corresponding to the LL-band and HP-bands as sequences of 1D Markov sources. Since each EOL symbol defines a known point, we then apply the first order Markov version of the MAP-BSC algorithm to the data between the EOL symbols. In this section, we will examine the four output streams to see if there is any residual redundancy in them that can be used by the MAP decoder.

Figures 5.17 and 5.18 compare the marginal probabilities of the symbols with the conditional probabilities (conditioned on the receipt of symbol indexed "1", which corresponds to the value 0) for the LL bands in the decomposition of "Lena" and "Barb" images. From these figures it can be seen that even though we applied a wavelet transform to the signal, the LL-band has residual redundancy in the form of memory, as is evidenced by the fact that the conditional probabilities are significantly different from the marginal probabilities. Calculating the probabilities conditioned on other values of indices is harder, since there are very few events (probabilities are very close to zero) in these cases. Also, the autocorrelation coefficient of the LL-band of the "Lenna" and "Barb" images were found to be 0.76 and 0.82 respectively, showing that the LL-bands are still heavily correlated.

Figures 5.19 through 5.24 show the marginal and conditional probabilities for the sub-sources 1 through 3 for the "Barb" and "Lenna" images. The conditional probabilities are again conditioned on the receipt of symbol indexed "1",

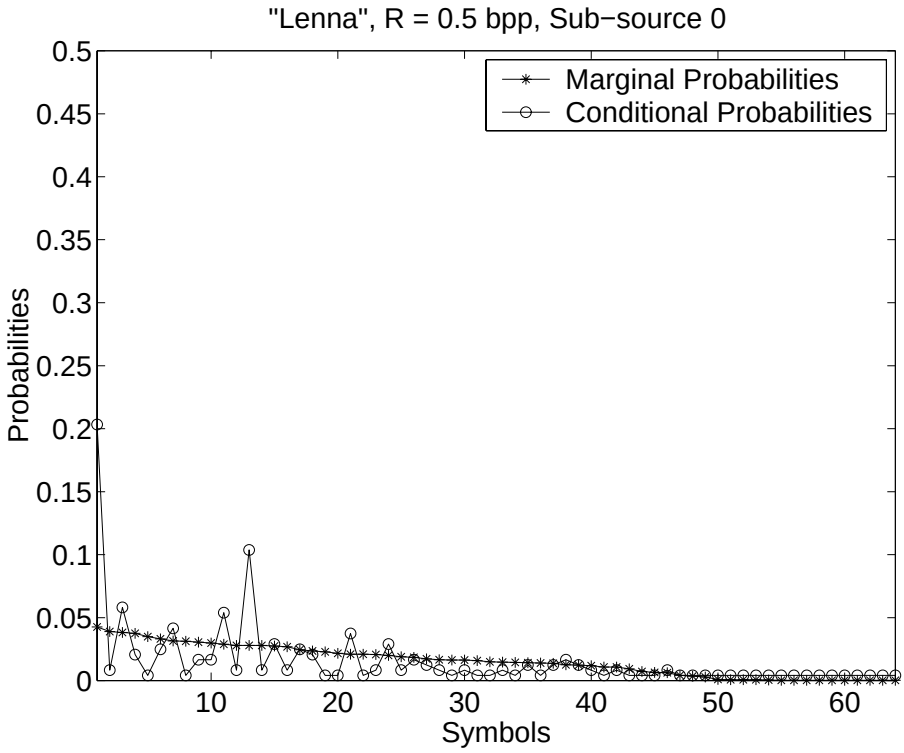


Figure 5.17: Comparison of the marginal and conditional probabilities for the LL-band in the "Lena" image, quantized at 0.5bpp.

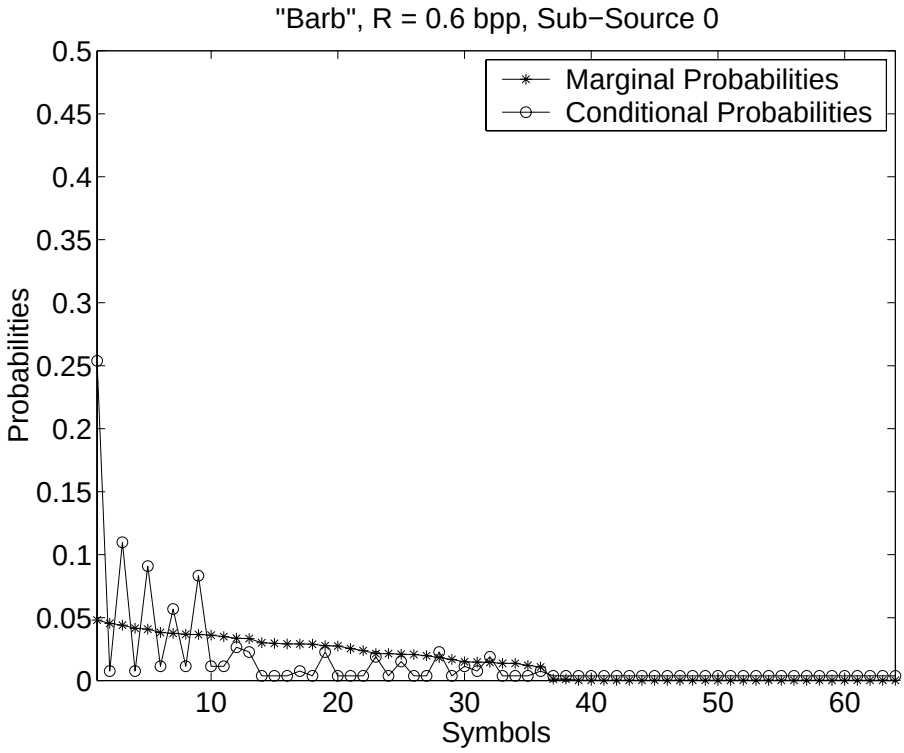


Figure 5.18: Comparison of the marginal and conditional probabilities for the LL-band in the "Barb" image, quantized at 0.6bpp.

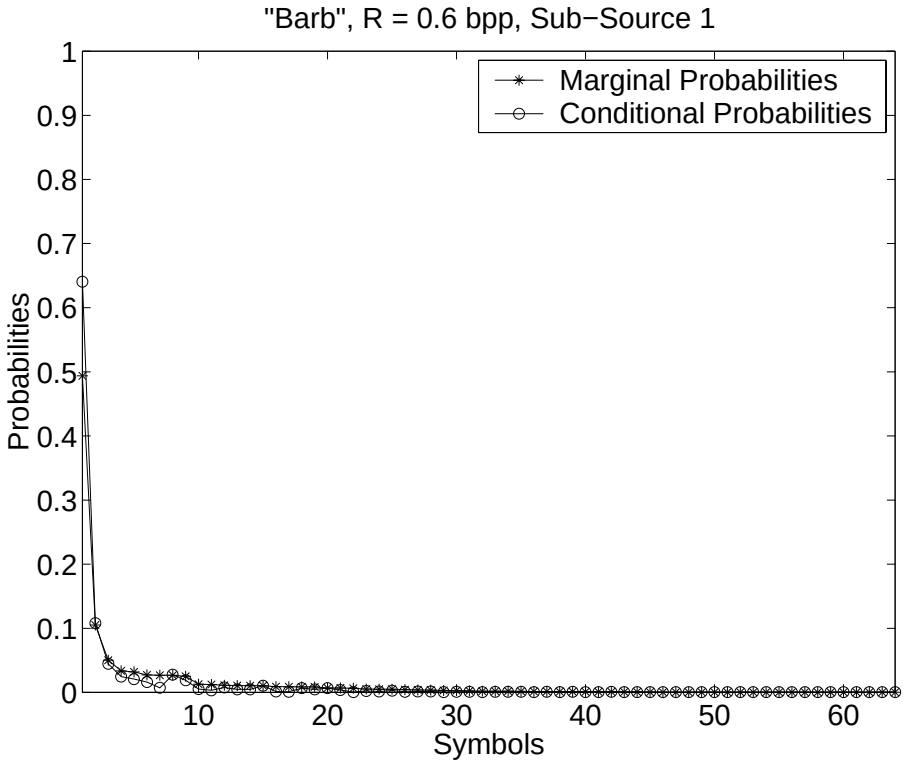


Figure 5.19: Comparison of the marginal and conditional probabilities for sub-source 1 of the "Barb" image, quantized at 0.6bpp.

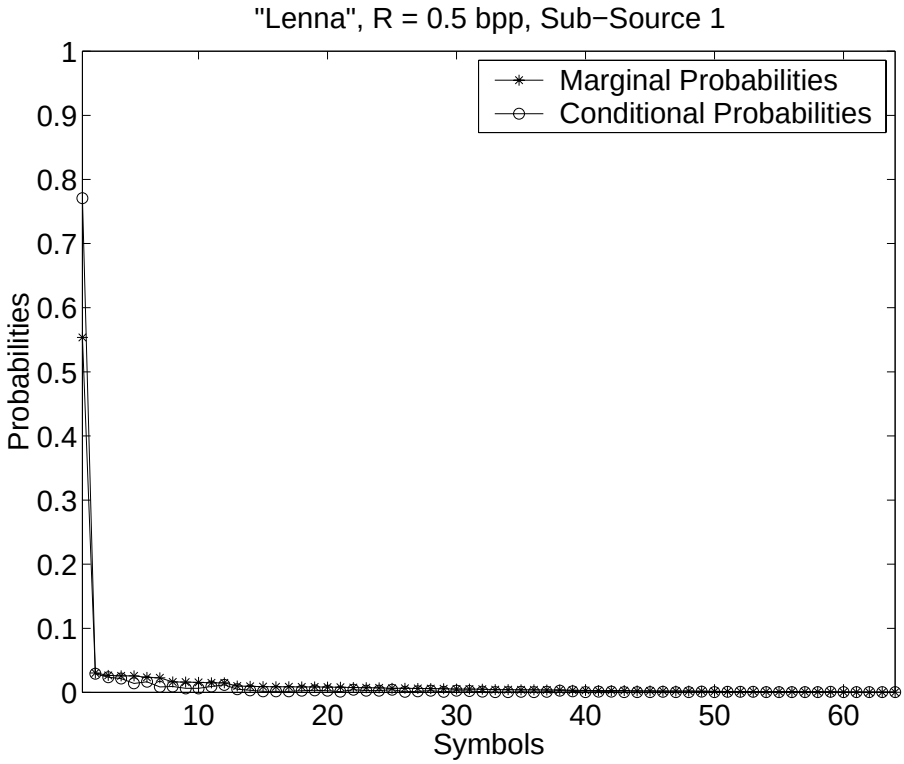


Figure 5.20: Comparison of the marginal and conditional probabilities for sub-source 1 of the "Lenna" image, quantized at 0.5bpp.

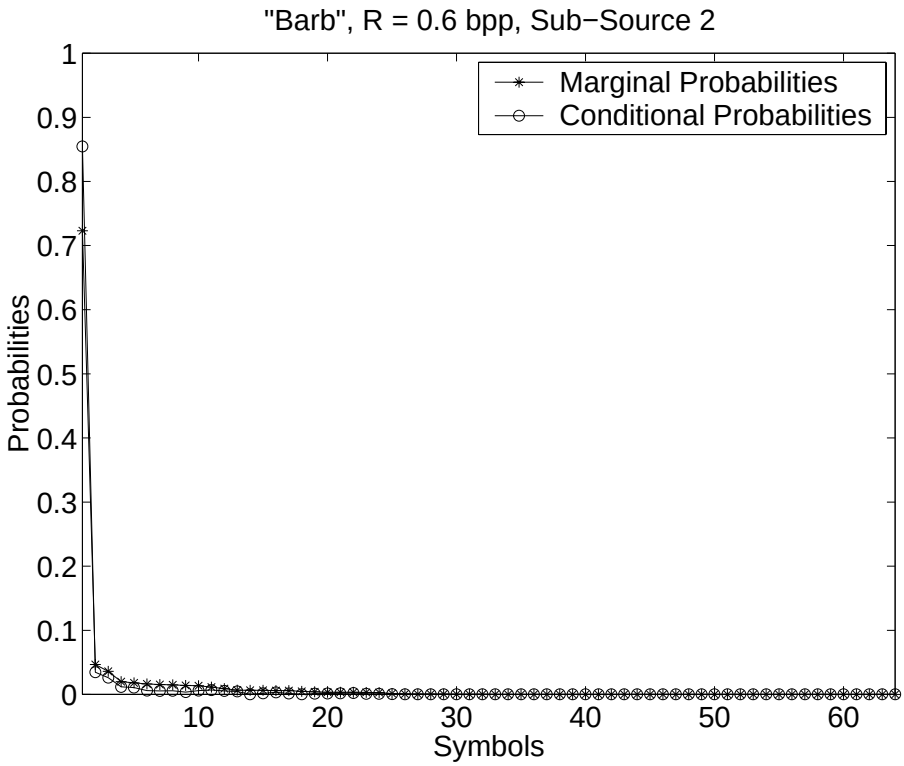


Figure 5.21: Comparison of the marginal and conditional probabilities for sub-source 2 of the "Barb" image, quantized at 0.6bpp.

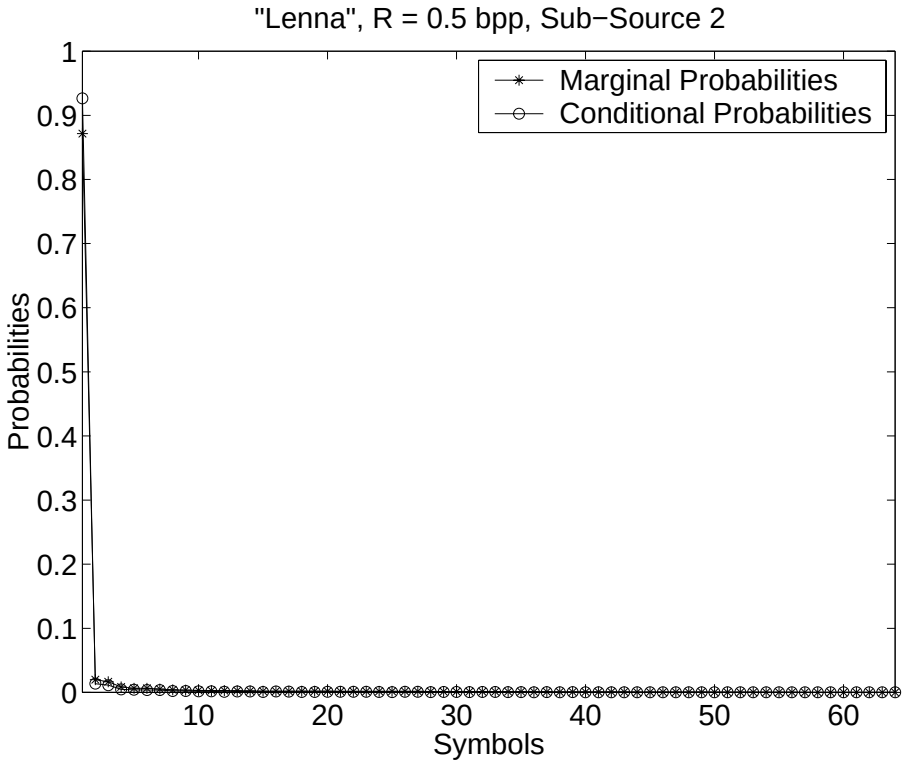


Figure 5.22: Comparison of the marginal and conditional probabilities for sub-source 2 of the "Lenna" image, quantized at 0.5bpp.

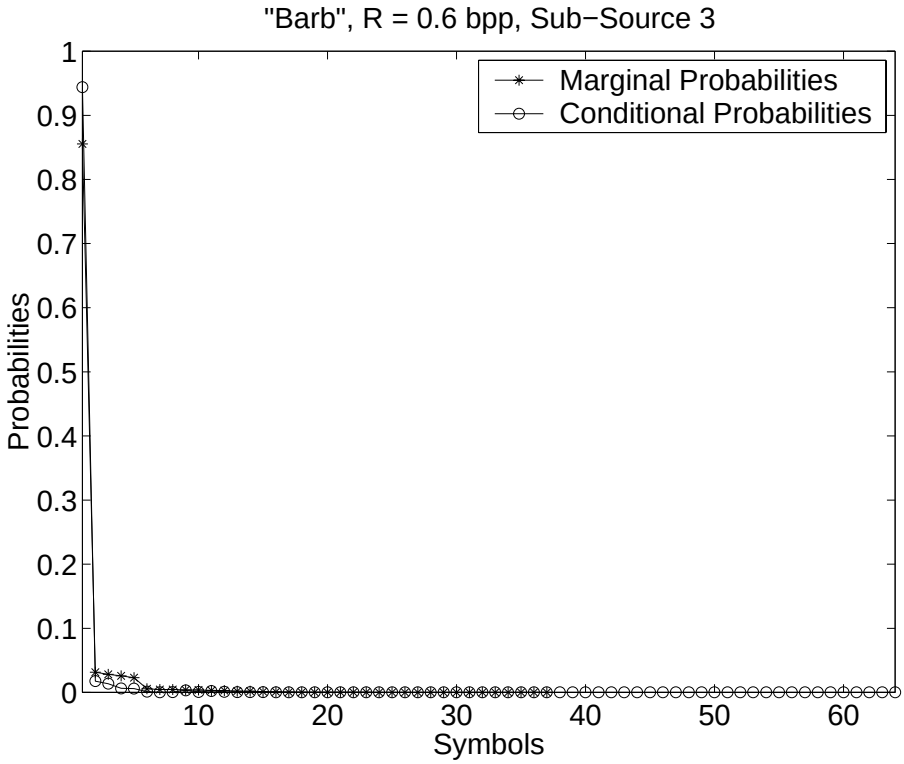


Figure 5.23: Comparison of the marginal and conditional probabilities for sub-source 3 of the "Barb" image, quantized at 0.6bpp.

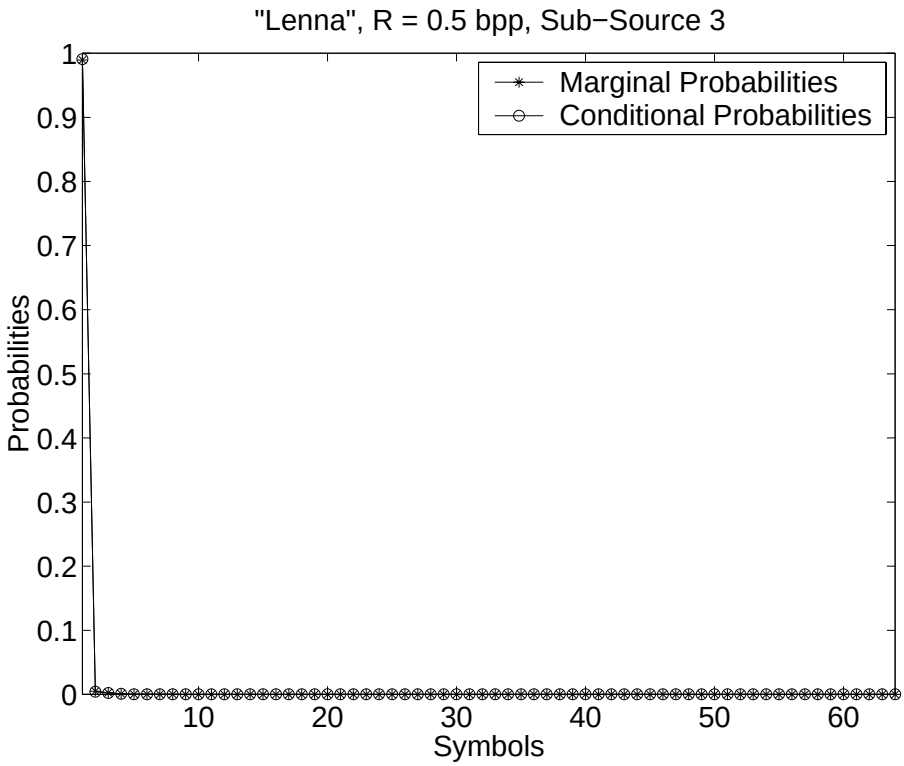


Figure 5.24: Comparison of the marginal and conditional probabilities for sub-source 3 of the "Lenna" image, quantized at 0.5bpp.

which corresponds to the vector $(0,0,0)$. As before, we see that there is residual redundancy in the form of memory in these sub-bands because the conditional and marginal probabilities are different for each of these sources, except for the sub-source 3 of the "Lenna" image. However, the fact that the marginal and the conditional probabilities are same does not imply that the source does not have memory, especially when the pdf has too many zero probability symbols in it. From these figures, we also observe that there is significant residual redundancy in the form of non-uniformity of the pdf as well. For these reasons, we expect (and will demonstrate in the next section) that the MAP-BSC decoder will perform better than a decoder that does not make use of these residual redundancies. Note that the memory in the image source is slightly different than that in an AR(1) source (test sources used in Chapter 3 and 4), where it was manifest as the autocorrelation of the source.

5.3 Results

The four output streams from the coder described in the above section are separately passed through a BSC with channel error rates, ϵ, ranging from 10^{-2} to 10^{-4}. The results presented in this section are an average of 5 channel realizations. PSNRs of the MAP decoded image is compared to that of the Huffman decoded image in Table 5.4 and Fig 5.25 for the 512×512 "Lena" image and error-bars are used to show the variation between the different channel realizations. As discussed before (Section 2.4), the MSNR is not an appropriate for images, because images are very sensitive to misplacement of even a single pixel. From the table and figure, it is seen

$log_{10}(\epsilon)$	Huffman Decoder PSNR in dB	MAP Decoder PSNR in dB
-2.0	16.14	20.11
-2.5	19.57	23.46
-3.0	23.93	27.60
-3.5	28.62	30.54
-4.0	29.83	31.06

Table 5.4: Comparison of PSNR Values for MAP decoded and Huffman decoded "Lenna" images for error rates between 10^{-2} and 10^{-4}

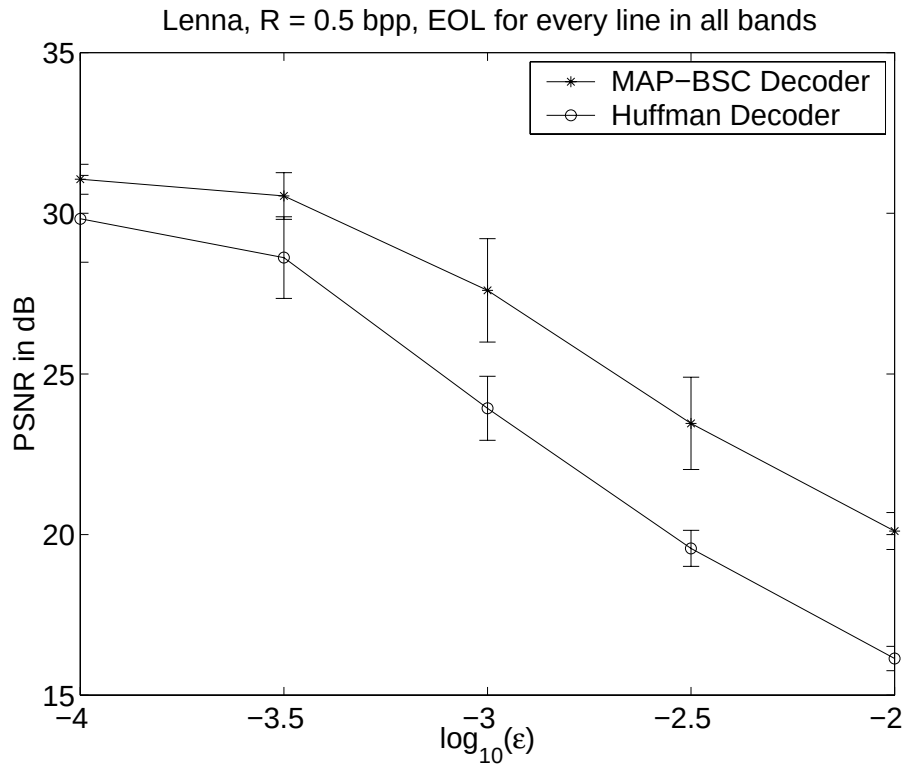


Figure 5.25: Comparison of the PSNR values of the MAP-BSC decoder and the Huffman decoder for the "Lena" image, quantized at 0.5bpp.

that MAP-BSC decoder does about 4dB better than the Huffman decoder at higher error rates and about 2dB better at lower error rates. To see the perceptual impact of this new approach, the Huffman decoded and the MAP-BSC decoded images are shown in Figure 5.26 and for $\epsilon = 10^{-2.5}$. Note that in these figures there are generally two kinds of error patterns. One is the bar-like error pattern where a rectangular bar of pixels is corrupted. The other kind is the ringing effect seen near the edges in the image. The square error patterns (as opposed to single pixel width disturbances) represent errors occuring in the LL-band. This happens only in the LL-band since the data in the LL-band is expanded upon reconstruction. The bar-like pattern also implies synchronization loss effect, since we are dealing with a BSC. A whole bar of pixels can be erroneously decoded only if synchronization was lost at the beginning of the bar. The ringing effects along the edges in the image are the effect of errors occuring in the HP-bands. From the figures it can also be seen that the MAP decoder is able to limit both the number of synchronization loss events as well as their length. Hence it can be seen that just as in the case of 1D signals (Chapter 3), the MAP decoder is able to perform well in terms of the number (and hence percentage) of bits out of synchronization as well as the PSNR.

The corresponding values for the "Barb" image are tabulated in Table 5.5 and plotted in Fig 5.27 for the same error rates. From these values, it is seen that

$log_{10}(\epsilon)$	Huffman Decoder PSNR in dB	MAP Decoder PSNR in dB
-2.0	15.22	18.79
-2.5	18.52	21.43
-3.0	22.86	24.82
-3.5	26.36	26.48
-4.0	28.08	27.91

Table 5.5: Comparison of PSNR Values for MAP decoded and Huffman decoded "Barbara" images for error rates between $10^{-2.0}$ and $10^{-4.0}$

the MAP-BSC decoder does better than the Huffman decoder at the error rates under consideration. The maximum improvement is about 3 dB at lower error rates. Figure 5.28 represent the Huffman decoded and the MAP-BSC decoded "Barbara" images at $\epsilon = 10^{-2.5}$, respectively and the trend in perceptual quality is very much

Huffman Decoded Image

MAP-BSC Decoded Image

Figure 5.26: Huffman and MAP decoded "Lenna" images at error rate $\epsilon = 10^{-2.5}$

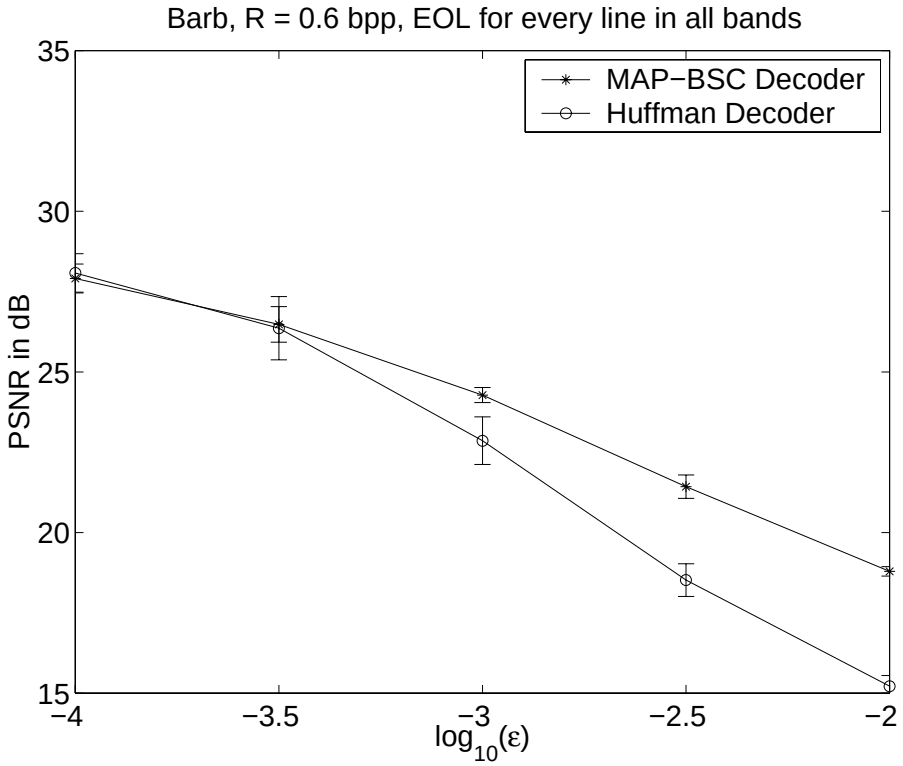


Figure 5.27: Comparison of the PSNR values of the MAP-BSC decoder and the Huffman decoder for the "Barb" image, quantized at 0.6bpp.

like in the case of the "Lenna" image.

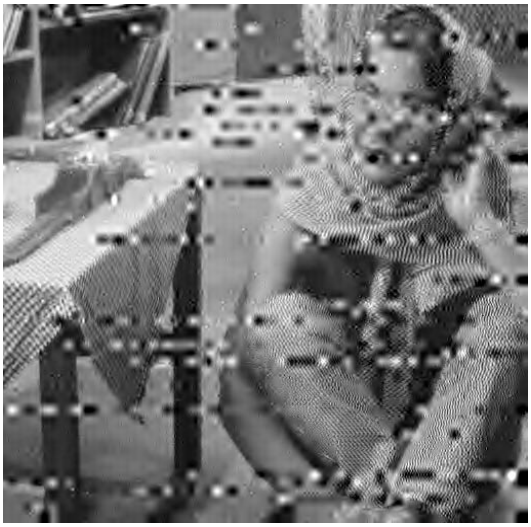
Since MAP-BSC decoder performs differently for the two images, we studied some of the properties of both the images that might have an influence on the performance of the decoder. These parameters are related to the lengths of the Huffman codewords, which can conceivably have an effect on the synchronization loss and recovery of the decoders. The parameters compared were: the minimum ($l_{\min}$) and maximum lengths ($l_{\max}$) in the codebook, the average length (l_{avg}) and the variance ($\sigma^2(l)$) of the lengths of the codewords in the codebook (all tabulated in Table 5.6). From this table it is

Source	**"Lenna"**				**"Barb"**			
	$l_{\min}$	$l_{\min}$	l_{avg}	$\sigma^2(l)$	$l_{\min}$	$l_{\min}$	l_{avg}	$\sigma^2(l)$
S_0	5	12	5.498	0.6248	4	13	5.1633	0.7569
S_1	1	12	3.3860	7.7783	1	13	3.2812	6.8444
S_2	1	12	1.629	3.0305	1	13	2.1756	4.0934
S_3	1	12	1.0289	0.1556	1	13	1.4725	1.7126

Table 5.6: Comparison of some parameters for the "Lenna" and "Barb" images

seen that there is not much difference in the values of any of the parameters for the "Barb" and "Lenna" images.

We next look at the residual redundancies present in the two images. As seen from the Figures 5.17 and 5.18, the conditional and marginal probabilities of the LL-bands of both the images bear similar relationships to the marginal probabilities. This result implies that we expect that the residual redundancies in the form of memory is similar in both the images. From Figures 5.19 through 5.23 and Figures 5.20 through 5.24 we see that the pdfs of sub-sources from "Lenna" image are peakier than those from the "Barb" image. Even though the differences between the conditional and marginal probabilities for the two images are different, these differences do not offer a concrete explanation for why the performance of the decoder is different for the two images. Hence it is not possible to conclusively state the reasons for this behaviour.



Huffman Decoded Image

MAP-BSC Decoded Image

Figure 5.28: Huffman decoded and MAP decoded "Barbara" images at error rate $\epsilon = 10^{-2.5}$

5.4 Conclusions

The MAP-BSC decoder developed in Chapter 3 was adapted to low bit-rate image transmission in this chapter, thereby testing it in a real world application. It was shown that the MAP-BSC offers significant improvements over the Huffman decoder. Even though we tested the algorithm on images, it is potentially applicable to 1D signals like speech, where synchronization loss does not have as debilitating an effect as it does in images and hence will require fewer resynchronization symbols. A sliding window approach may be used to take care of the non-stationarity of the signal.

The decoder uses the residual redundancy present along only one dimension of the image. One of the interesting directions in which to continue the work presented in this chapter, would be to develop a decoder that uses the residual redundancy in the image along both, the horizontal and the vertical dimensions. Also, the performance of the codec presented in this chapter may be improved by forming larger vectors in the higher frequency subbands. We noted that the run-length coding in the JPEG scheme removes a lot of the redundancy due to memory, there may yet be some redundancy due to non-uniformity of pdf and it may be interesting to study this redundancy and the potential applicability of the MAP-BSC decoder to the JPEG coder. It would also be a challenging problem to development of a coder that makes use of the dependence between the subbands and which can be used along with the MAP decoders in presented in this dissertation.

Chapter 6

Conclusions

The observation that Shannon's source-channel separation theorem (1948) holds only under asymptotic conditions led to the popularity of joint source-channel coding (JSCC) schemes as alternative error-resilience measures, especially for transmission over channels with severe bandwidth constraints. Joint source-channel decoders (JSCD) are JSCC techniques that concentrate on the design of the channel and the source decoder in some joint sense. JSCDs typically aim at using the residual redundancy in the encoded source after source coding. JSCDs have been designed for fixed-length encoded sources and can be found in the literature. In this dissertation, we designed optimal JSCDs for variable-length encoded sources over channels with and without memory. This technique was then applied to a subband coding based image codec that used entropy coding extensively.

6.1 Summary and Original Contributions

In a noisy environment, a decoder that searches over all possible sequences to find the "best" estimate of the transmitted sequence can be expected to do better than a decoder that does not anticipate any errors in the sequence. This search, however, is time consuming and the computational burden can be substantially reduced by casting the problem in a dynamic programming format, where some obviously "bad" candidates can be eliminated in the early stages of the algorithm. For fixed-length encoded sources, the bits can be processed one word at a time, since the word boundaries are fixed. However, this is not a good solution when variable-length codes are

used and a new state-space structure is needed. This dissertation proposes such a state-space structure and allows us to deal with the variable-length nature of the codes in an elegant manner. The algorithms proposed in this dissertation are optimal in the sense of finding the maximum a posteriori probability (MAP) sequence.

In Chapter 3, we proposed one of the first solutions to the MAP decoding problem for Markov sources over binary symmetric channels and then specialized it to sources without memory. Experiments were conducted to compare the performance of the proposed decoders to the Huffman decoders and it was shown that the proposed decoder (for sources with memory) performed significantly better over a range of channel error rates and source memory in terms of both MSNR and PBOS. It was shown that the improvement offered by the proposed decoder increased with an increase in the memory of the source. The maximum improvement in MSNR over the Huffman decoder was found to range from about 8 dB (for an AR(1) source with $\rho_s = 0.9$, quantized to 3 bits per sample) to approximately zero for an almost uncorrelated source ($\rho_s = 0.3$). It was also shown that the complexity of the decoder can be reduced if the greatest common divisor of the lengths in the codebook is greater than one.

The algorithm was then tested over various channel mismatch conditions. These tests were motivated by the fact that the channel statistics may not be estimated accurately and such tests will reveal the robustness of the decoder to channel mismatches. It was found that the MAP-BSC decoder is robust to these inaccuracies. So long as the estimated error rate remains less than 2-orders of magnitude lower than the actual error rate, the performance degradation was well below 3 dB relative to the perfectly matched MAP-BSC. Despite such severe mismatch conditions, the MAP-BSC out did the Huffman decoder for most error rates.

The MAP-BSC decoder did not offer any significant improvement for the memoryless source considered here, because the pdf of this source is not very peaky and hence the residual redundancy in this source is rather weak. Testing the algorithm for memoryless sources with peakier distribution is left for future work.

Signals like speech are often modeled as higher order auto-regressive processes and hence it is also of interest to modify the algorithm for higher order Markov sources. Chapter 3 also provides a generalization of the MAP decoding problem formulation for Markov sources of higher orders.

Many real-life channels have memory and even though it may be possible to render them essentially memoryless by interleaving the data, this causes delays which may be unacceptable in some situations. Hence, it is of interest to design JSCD decoders for channels with memory. In Chapter 4, we proposed the design of the optimal MAP decoder for channels with memory for the first time. This decoder uses the additive Markov channel (AMC) model for the channel. It was shown that the state space developed for the MAP-BSC for Markov sources can also be used in the dynamic programming formulation for the MAP-AMC. The difference between the MAP-AMC and the MAP-BSC was mainly in the metric, which in the case of MAP-AMC involves channel transition probabilities as well. This property implies that the algorithm should remember if the previous bit was erroneous at each stage of the algorithm. A discussion was given to show that the complexity of the MAP-BSC and the MAP-AMC are nearly the same even though the number of operations at each stage is higher for the MAP-AMC.

Simulation results showed that the MAP-AMC decoder does significantly better than the conventional decoder at all channel error-rates, in terms of both MSNR and PBOS. The maximum improvement for an AR(1) source with $\rho_s = 0.9$ quantized at a rate of 3 bits per sample was about 6 dB for a channel with correlation coefficient, ρ_c, equal to 0.8. Experiments were conducted to see the effect of lowering the channel correlation coefficient on the relative performance of the MAP-AMC decoder over the Huffman decoder. It was seen that the performance improvement that the MAP-AMC offers over the Huffman decoder increases as the value of ρ_c decreases. Experiments conducted on a memoryless source showed that the MAP-AMC performed essentially the same as the Huffman decoder. As before, experiments on sources with peakier pdfs might prove to be an interesting avenue in which to direct future work.

Robustness studies were conducted on this decoder which showed that the decoder was robust to inaccuracies in the measurement of channel correlation and only a huge mis-match of 0.9 caused the MAP-AMC result to drop below the performance of the Huffman decoder.

Although the proposed MAP-BSC decoder was tested on some synthetic sources in the third and fourth chapters, it is interesting to apply the algorithms to more practical situations where the source cannot be modeled as an AR(1) source. An example of such a source is an image, where the subband coefficients are not necessarily

AR(1). In Chapter 5, we discussed the application of the MAP-BSC decoder to such a scenario, for the first time. Here the memory is not present in the form of a correlation coefficient but is manifested as a difference in the conditional and marginal pdfs of the source symbols. A source codec that performs nearly as well as the baseline JPEG codec was also designed in this chapter. It was shown that this codec generates sources that have significant residual redundancy in the form of both memory and non-uniformity of pdf. Experiments showed that the MAP-BSC algorithm performed significantly better compared to the Huffman decoder both perceptually and in terms of an objective PSNR. For the "Lenna" image, the improvement ranges from 2 to 4 dB and for the "Barb" image the maximum improvement is 3 dB.

6.2 General Conclusions

One of the salient features of the algorithms presented in this dissertation is the fact that the complexity of the decoders does not increase with time. This makes it possible to find the exact MAP sequence in finite time, on an average. Also, the decoders do not assume the knowledge of the number of bits or words that were transmitted. This dissertation introduced a new state-space structure that could potentially be used in other applications dealing with variable-length codes. Examples of such applications are the design of joint source-channel *encoders* that use entropy coding, like channel constrained ECVQs.

The proposed decoders are delayed-decision decoders which means that they may not be applicable to video signals, where there are stringent restrictions on the permissible delays. Also, the fact that these decoders need the source statistics to operate implies that making them adaptive can be expensive in terms of the overhead that will have to be incurred in refining the source statistics.

For good performance improvement over the conventional decoders, there must be residual redundancy in the source. Some coders tend to reduce these redundancies (Sections 5.1.1 and 2.5.3) and in such situations a compromise will have to be struck between the compression efficiency of the encoder and the amount of residual redundancy in the source before the proposed decoders can be used.

6.3 Future Work

The work done in this dissertation could be extended along a few interesting directions, some of which are direct applications and some use the ideas presented in this dissertation as starting points. One example of the first kind would be to apply the MAP-AMC decoder to image signals along the lines of the MAP-BSC decoder.

It was noted that run-length coding in the JPEG scheme removes a lot of the redundancy due to memory. However, there could potentially be some residual redundancy both in the form of memory and in the form of a non-uniform pdf in a JPEG encoded image. It would be interesting to study this redundancy and the potential applicability of the MAP decoders to the JPEG coder.

Also, in Section 2.5.3, we briefly touched upon the conflicting requirements between the arithmetic coder and the MAP decoder. It was argued that the efficiency of the arithmetic coder increases on encoding long sequences of symbols which would reduce the correlation between the various sub-strings. However, it is conceptually possible to trade-off arithmetic coding efficiency for residual redundancy to achieve better R-D performance under noisy conditions. Studying this would be another interesting direction in which to pursue the work presented in this dissertation.

As mentioned earlier, speech signals are often modeled as higher order AR processes. The higher order extension to the algorithms presented in this dissertation may be tested on speech signals.

The JSCDs presented in this dissertation make a "hard" decision on the codewords that were used to reconstruct the signal. It is also possible to pick the quantizer output points so as to minimize the mean squared error of the reconstructed signals under noisy conditions. Such JSCDs are called minimum mean squared error (MMSE) decoders. MMSE decoders have been designed for fixed-length encoded sequences. It is potentially possible to extend the work presented in this dissertation to develop an MMSE decoder for variable-length encoded sources.

Since channel codes can be used to further enhance the performance of a system under noisy conditions, it is natural to design *integrated JSCDs* as a counter-part to the integrated JSCEs mentioned in Section 2.2. Since convolutional codes have a state-space associated with them, they are a natural choice for the channel codes that can be used in conjunction with the MAP decoders presented in this dissertation. This

is so because it is potentially possible to combine the state-space of the convolutional encoder with that of the MAP decoders presented in this dissertation.

The proposed state-space could potentially be used to design joint source-channel *encoders* for specific types of source codecs. As mentioned before, channel constrained ECVQ is one example of such a codec.

The MAP-BSC decoders when adapted to image transmission, used the residual redundancy only along the horizontal direction. MAP decoders could potentially be developed that can use the residual redundancies in both the directions of a 2-D source; however, this may require the development of a new state-space structure.

It would also be a challenging problem to develop a coder that makes use of the dependence between the subbands and which can be used along with the MAP decoders presented in this dissertation.

Although the image codec developed here was not state-of-the-art, it is a reasonably sophisticated codec and the results obtained in this dissertation might imply that the MAP-BSC algorithm could potentially be a viable alternative for achieving error-resilience in more sophisticated codecs. To conclude, the ideas in this dissertation could be used as stepping stones for future work in the broader area of joint source-channel coding.

References

Abramson, N. (1963). *Information Theory and Coding.* McGraw-Hill, New York.

Alajaji, F., N. Phamdo, N. Farvardin, and T. Fuja (1996, January). Detection of binary Markov sources over channels with additive Markov noise. *Transactions on Information Theory 42*(1), 230–239.

Antonini, M., M. Barlaud, and P. M. I. Daubechies (1992). Image coding using wavelet transform. *IEEE Transactions on Image Processing 1*, 205–220.

Blahut, R. E. (1987). *Principles and Practice of Information Theory.* Reading, Massachusets: Addison-Wesley.

Burlina, P., F. Alajaji, and R. Chellappa (1996). *Technical Report CAR-TR-814* .

Burrus, C., R. Gopinath, and H. Guo (1998). *Introduction to Wavelets and Wavelet transforms - A Primer.* Upper Saddle River, New Jersey: Prentice Hall.

Chou, P., T. Lookabaugh, and R. Gray (1989, January). Entropy constrained vector quantization. *IEEE Transactions on Accoustics, Speech and Signal Processing* , 31–42.

Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math 41*, 909–996.

Daubechies, I. (1992). *Ten Lectures on Wavelets.* Philadelphia, PA: SIAM.

Demir, N. and K. Sayood (1998). Joint source/channel coding for variable length codes. In *IEEE Data Compression Conference*, Snowbird, Utah, pp. 139–148.

Dunham, J. G. and R. M. Gray (1981, July). Joint source and noisy channel trellis encoding. *IEEE Transactions on Information Theory 27*(4), 516–519.

Equitz, W. H. (1989, October). An new vector quantizer clustering algorithm. *IEEE Transactions on Accoustics, Speech and Signal Processing* , 1568–1575.

Farvardin, N. (1990, July). A study of vector quantization for noisy channels. *IEEE Transactions on Information Theory 36*(4), 799–809.

Farvardin, N. and J. W. Modestino (1984). Optimum quantizer performance for a class of non-gaussian memoryless sources. *IEEE Transactions on Information Theory 30*, 485–497.

Ferguson, T. J. and J. H. Rabinowitz (1984, July). Self-synchronizing Huffman codes. *IEEE Transactions on Information Theory IT-30*(4), 687–693.

Gersho, A. and R. M. Gray (1992). *Vector Quantization and Signal Compression*. Kluwer Academic Publishers.

Gibson, J., T. Berger, T. Lookabaugh, D. Lindebergh, and R. L. Baker (1998). *Digital Compression for Multimedia Principles and Standards*. San Francisco, California: Morgan Kauffman Publishers, Inc.

Ho, K.-P. and J. M. Kahn (1996, November). Transmission of analog signals using multicarrier modulation: A combined source-channel coding approach. *IEEE Transactions on Communication 44*(11), 1432–1443.

Hochwald, B. and K. Zeger (1997). Trade off between source and channel coding. *IEEE Transactions on Information Theory 43*, 1412–1424.

Huang, J.-Y. and P. Schultheiss (1963, September). Block quantization of correlated Gaussian random variables. *IEEE Transactions on Communications 11*, 289–296.

Huffman, D. (1951). A method for the construction of minimum redundancy codes. *Proceedings of the IRE 40*, 1098–1101.

Huffman, D. (1952, January). A method for construction of minimum redundancy codes. *Proc. IRE 40*, 1098–1101.

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

Jelinek, F. (1968). *Probabilistic Information Theory.* McGraw-Hill, New York.

Lai, W.-M. and S. Kulkarni (1996, May). Extended synchronization codewords for binary prefix codes. *IEEE Transactions on Information Theory 42*, 984–987.

LeGall, D. (1991, April). MPEG: A video compression standard for multimedia applications. *Communications of the ACM 34*(4), 46–58.

Lei, S.-M. and M.-T. Sun (1991, March). An entropy coding system for digital hdtv applications. *IEEE Transactions on Circuits and Systems for Video Technology 1*(1), 147–154.

Linde, Y., A. Buzo, and R. M. Gray (1980, January). An algorithm for vector quantizer design. *IEEE Transactions on Communications 28*, 84–95.

Llados-Bernaus, R. and R. L. Stevenson (1998, October). Fixed-length entropy coding for robust video compression. *IEEE Transactions on Circuits and Systems for Video Technology 8*(6), 745–755.

Mallat, S. G. (1989, July). A theory of multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-11*, 674–693.

Malvar, H. and D. H. Staelin (1989, April). The LOT:transform coding without blocking effects. *IEEE Transactions on Accoustics, Speech and Signal Processing 37*, 553–559.

Melsa, J. and D. Cohn (1978). *Decision and Estimation Theory.* McGraw Hill.

Modestino, J. W., D. G. Daut, and A. L. Vickers (1981, September). Combined source channel coding of images using the block cosine transform. *IEEE Transactions on Communication 29*, 1262–1274.

Park, M. and D. J. Miller (1997, October). Low-delay optimal MAP state estimation in HMMS with application to symbol decoding. *IEEE Signal Processing Letters 4*(10), 289–292.

Park, M. and D. J. Miller (1998a, March 18-20). Decoding entropy-coded symbols over noisy channels by MAP sequence estimation for asynchronous HMMs. In *Conference on Information Science and Systems*, Princeton, New Jersey, pp. 477–482.

Park, M. and D. J. Miller (1998b, February). A sequence-based, approximate MMSE decoder for source coding over noisy channels using discrete hidden Markov models. *IEEE Transactions on Communications 46*(2), 222–231.

Park, M. and D. J. Miller (1999, March 15-19). Joint source-channel decoding for variable-length encoded data by exact and approximate MAP estimation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, Phoenix, Arizona, pp. 2451–2454.

Pasco, R. (1976). *Source coding algorithms for data compression.* Ph. D. thesis, Stanford University.

Pennebaker, W. and J. Mitchell (1993). *JPEG still image compression standard.* Van Norstrand Reinhold, New York.

Phamdo, N. (1993). *Quantization Over Discrete Noisy Channels Under Complexity Constraints.* PhD Dissertation, University of Maryland.

Phamdo, N., F. Alajaji, and N. Farvardin (1997, January). Quantization of memoryless and gauss-Markov sources over binary Markov channels. *IEEE Transactions on Communication 45*(6), 668–674.

Phamdo, N. and N. Farvardin (1994, January). Optimal detection of discrete Markov sources over discrete memoryless channels. *IEEE Transactions on Information Theory 40*, 186–193.

Phamdo, N., N. Farvardin, and T. Moriya (1993, May). A unified approach to Tree-Structured and Multi-Stage vector quantization for noisy channels. *IEEE Transactions on Communications 39*(3), 835–850.

Poor, H. V. (1988). *An Introduciton to Signal Detection and Estimation.* Springer-Verlag.

Rao, K. and J. J. Hwang (1996). *Techniques and Standards for Image.Video and Audio Coding*. Prentice Hall PTR, Upper Saddle River, New Jersey.

Rao, K. and P. Yip (1990). *Discrete cosine transform : algorithms, advantages, and applications*. Boston: Harcourt Brace Jovanovich.

Redmill, D. W. and N. G. Kingsbury (1996, April). The EREC: An error-resilient technique for coding variable-length blocks of data. *IEEE Transactions on Image Processing 5*(4), 565–574.

Riskin, E. A. (1991, March). Optimal bit allocation via the generalized bfos algorithm. *IEEE Transactions on Information Theory 37*(2), 400–402.

Rissanen, J. (1979, May). Generalized Kraft inequality and arithemtic codes. *IBM Journal of Research and Development 20*, 198–203.

Rissanen, J. and G. Langdon (1981). Arithmetic coding. *IBM Journal of Research and Development 23*(2), 149–162.

Rydbeck, N. and C. W. Sundberg (1976, January). Analysis of digital errors in nonlinear PCM. *IEEE Transactions on Communication 24*, 59–65.

Said, A. and W. A. Pearlman (1996, June). A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on circuits and systems for video technology 6*(3), 243–250.

Sayood, K. (1996). *Introduciton to Data Compression*. San Francisco, California: Morgan Kaufmann Publishers Inc.

Sayood, K. and J. C. Borkenhagen (1991, June). Use of residual redundancy in the design of joint source channel coders. *IEEE Transactions on Communications 39*(8), 839–846.

Shannon, C. (1959, March). Coding theorems for a discrete source with a fidelity. *IRE Nat. Conf. Rec.* , 142–163.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell Systems Technical Journal 27*, 379–423 and 623–656.

Shapiro, J. M. (1993, December). Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing 41*, 3445–3462.

Shoham, Y. and A. Gersho (1988, September). Efficient bit allocation for an arbitraty set of quantizers. *IEEE Transactions on Accoustics, Speech and Signal Processing 36*(9), 1445–1453.

Strang, G. and T. Nguyen (1997). *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Massachussets.

Subbalakshmi, K. P. and J. Vaisey (1998). Optimal decoding of memoryless sources over binary symmetric channels. In *IEEE Data Compression Conference*, Snowbird, Utah, pp. 573.

Subbalakshmi, K. P. and J. Vaisey (1999a, June). Joint source-channel decoding of entropy coded Markov sources over Binary Symmetric Channels. In *IEEE International Conference on Communications*, Volume 1, Vancouver, Canada, pp. 446–450.

Subbalakshmi, K. P. and J. Vaisey (1999b, March). Optimal decoding of entropy coded Markov sources over channels with memory. In *The 33rd Annual Conference on Information Sciences and Systems*, Baltimore, Maryland.

Taubman, D. High performance scalable image compression with EBCOT. To be published in IEEE Transactions on Image Processing.

Titchener, M. (1997, March). The synchronizaiton of variable-length codes. *IEEE Transactions on Information Theory 43*(2), 683–691.

Vaidyanathan, P. (1987, July). Quadrature mirror filter banks, M-band extensions and perfect reconstruction techniques. *IEEE ASSP Magazine 4*(3), 4–20.

Vaidyanathan, P. (1993). *Multirate Systems and Filter Banks*. Signal Processing Series. Englewood Cliffs, NJ 07632: Prentice Hall.

Vaisey, J., M. Barlaud, and M. Antonini (1998). Multispectral image coding using lattice VQ and the wavelet transform. In *IEEE International Conference on Image Processing*, Chicago, pp. 00–10.

Vembu, S., S. Verdu, and Y. Steinberg (1995, January). The source-channel separation theorem revisited. *IEEE Transactions on Information Theory 41*, 142–163.

Vetterli, M. and J. Kovacevic (1995). *Wavelet and Subband Coding*. Englewood Cliffs, NJ, Prentice Hall PTR.

Wallace, G. K. (1992, February). The JPEG still image compression standard. *IEEE Transactions on Consumer Electronics 38*(1), 28–33.

Wen, J. and J. D. Villasenor (1997). A class of reversible variable length codes for robust image and video coding. In *IEEE International Conference on Image Processing*, Santa Barbara, California, pp. 65–68.

Westerink, P., J.Biemond, and D. E. Boekee (1988). An optimal bit allocation algorithm for sub-band coding. In *IEEE International Conference on Accoustics, Speech and Signal Processing*, pp. 757–760.

Woods, J. (Ed.) (1991). *Subband Image Coding*. Signal Processing Series. Boston: Kluwer Academic.

Zeger, K. and A. Gersho (1990, December). Pseudo-gray coding. *IEEE Transactions on Communications 38*(12), 2147–2157.

Zeger, K., J. Vaisey, and A. Gersho (1992, February). Globally optimal vector quantizer design using stochastic relaxation. *IEEE Transactions on Signal Processing 40*, 310–322.