The Pennsylvania State University

The Graduate School

Department of Electrical Engineering

HIGH ORDER JOINT SOURCE CHANNEL DECODING AND CONDITIONAL ENTROPY ENCODING: NOVEL BRIDGING TECHNIQUES BETWEEN PERFORMANCE AND COMPLEXITY

A Thesis in

Electrical Engineering

by

Yu-wei Wang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2004

The thesis of Yu-wei Wang has been reviewed and approved* by the following:

David J. Miller
Associate Professor of Electrical Engineering
Thesis Adviser
Chair of Committee

John F. Doherty
Associate Professor of Electrical Engineering

John J. Metzner
Professor of Electrical Engineering

Natarajan Gautam
Associate Professor of Industrial Engineering

W. Kenneth Jenkins
Professor of Electrical Engineering
Head of the Department of Electrical Engineering

*Signatures are on file in the Graduate School

Abstract

Joint source-channel decoding techniques capitalize on residual redundancy that typically remains following a source encoding operation. These methods, which include MAP and MMSE-based decoders, estimate the sequence of encoded source symbols based on statistical knowledge of both the channel and the encoded source. Generally, these techniques are based on a Markov model for the quantized source and, thus, on a hidden Markov model for the source-channel tandem. The number of states in the hidden Markov model, and thus the computational and storage complexities, grow *exponentially* with the order (K) of the Markov model, i.e., the complexity order is $O(N^{K+1}T)$, with N the number of source quantization levels and T the length of the data sequence. Thus, to retain implementable complexity, low order models (K = 1,2) are typically used, at the expense of model accuracy. In this thesis, we propose methods to *bridge* the performance-complexity gap, i.e. to provide solutions that give better performance than a low order decoder while incurring only modest increases in complexity. We first propose using improved iterative scaling(IIS) algorithm to learn the maximum entropy a posteriori probability from a large group of pairwise constraints. This significantly reduces the training set size required in direct model learning via frequency counts to avoid the overfitting problem. However, the high computational complexity is still unavoidable with high order model. Therefore, we suggest using neighboring noisy symbols in conditioning to incur only modest computational complexity increase compared to a low order model. We further applied the idea of using IIS to approximate high order

conditioning in conditional entropy constrained encoding context. We then propose a second approach. This second decoding approach consists of two stages: 1) low order JSC decoding, followed by 2) a linear FIR filtering of the JSC decoded signal. The linear filter is chosen to provide an optimal (least squares) estimate of the original source. This approach provides an approximate way to increase the effective order of the decoder, yet while retaining quite manageable complexity. The new approach is demonstrated to significantly improve upon standard MMSE-based JSC decoding performance, both for the case of nonpredictive source coding (e.g. vector quantization) as well as for predictive source coding (DPCM). In our third approach, we tried to model the source sequence as a hidden Markov model whereas a Markov model is commonly used. This approach provides a source model which achieves infinite memory while a Markov source model only has finite memory. This translates to lower source entropy and better decoding performance compared to a SAMMSE decoder with comparable memory complexity, although our computational complexity is higher. As an extension, we also tried to model the source data as Markov random field with our sequence based mean field annealing method as the joint source channel decoder. This method relaxes the independence between pixels assumptions in the standard mean field annealing by assuming the independence is only between the rows of a image. By introducing more dependency between symbols into our source model, we further utilize this dependency to combat the noisy channel.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to express my special appreciation to my thesis advisor, Dr. David J. Miller, for the ideas, guidance, patience, and encouragement he has given me over the years of working with him. He has also shown me the true value of research, and sparkled my interest and curiosity on this subject in many ways. I will definitely carry this value with me into my future work. I would also like to thank all other commitee members, Dr. Doherty, Dr. Metzner, and Dr. Gautam for their insightful commentary on my work.

I would like to dedicate this thesis to my wife who has always shared my ups and downs in my work, and will continue to do so in my life. This is an achievement for the both of us.

Chapter 1

Introduction

1.1 Problem Statement

In recent years, many researchers have proposed joint source channel (JSC) coding systems in place of standard separately designed source and channel coders. The purpose of the source coder is to reduce redundancy in order to achieve maximum compression. The minimum rate at which the source can be losslessly communicated is the source entropy. The purpose of the channel coder is to add redundancy, which increases bit rate, in order to combat a noisy channel. Shannon has proved that the source bit rate can be reduced down to its entropy rate, and transmitted at that rate without any loss of information. However, to design an optimal source coder operating at the entropy rate usually comes with high computational complexity and a large amount of memory. A low complexity suboptimal source coder would produce encoded source symbols that retain statistical redundancies. A joint source channel decoder is designed to utilize this statistical redundancy to combat noisy channel conditions. This design can in some cases be used without an explicit channel coder and still achieve good decoding performance in noisy channel conditions. Sayood and Borkenhagen[54] coined the term residual redundancy which refers to this statistical dependency between encoded source symbols. There are three fundamental joint source channel (JSC) decoder structures. The first method is a sequence MAP decoder[48] which finds the decoded symbol sequence with

maximum *a posteriori* probability given the received noisy symbol sequence. The second method is a symbol MAP decoder[45] which finds the decoded symbol with the maximum *a posteriori* probability at each symbol time given the received noisy symbol sequence. In both sequence MAP and symbol MAP decoders, a lookup table is needed. This lookup table provides an estimation of $E[x_t|I_t = \hat{i}_t]$, which can be obtained from the training set using a centroid rule, where x_t denotes source value at time t and I_t denotes the random variable of the decoded source symbol index $\hat{i}_t$ at time t. This table gives a one to one mapping from the decoded indices to a finite set of reconstruction values. The third method is a minimum mean squared estimation (MMSE) decoder, which finds the *a posteriori* probability of the decoded symbol given the received symbol sequence, and then uses a conditional mean estimator[40] to estimate the transmitted quantization levels. By doing so, this method effectively creates an infinite number of reconstruction values as opposed to the previous two methods. All of these three methods model the source and channel tandem as a hidden Markov model at the decoder. The performance of all these decoding systems improve with increasing order of the hidden Markov model[49]. However, the memory storage and computational complexity increases *exponentially* with model order increase. Increasing model order also means more training data is needed for learning when training the model. For these reasons, to our knowledge no one has simulated these methods directly using a hidden Markov model with an order greater than (say) two or three.

In this thesis, we propose novel techniques which bridge the gap of complexity and performance trade-off in JSC decoding, i.e. to to provide solution that is better than low order decoder while incurring only a modest increase in complexity. One of our bridging

techniques uses high order conditioning context with low computational complexity and memory storage, *without* needing a large training data set, to ensure the quality of learning the model. We further applied this technique to high order conditional entropy constrained encoding to reduce the *large* training set data support needed from directly learning its statistical model via frequency counts. Our second bridging technique is a two stage estimation approach, which modifies the decoder structure to include additional linear parameters, with these parameters optimized to improve the decoder performance. These linear parameters are trained using a training set, and used to provide a heuristic way to increase the decoder order with small computational complexity increase. This new approach will be demonstrated to significantly improve upon low order standard MMSE-based JSC decoding performance. In our third bridging technique, we propose modeling the source symbols as a hidden Markov model. In previous works of JSC encoding/decoding, many researchers have modeled the encoded source symbols as a Markov model, and the source and channel tandem as a hidden Markov model[13, 23, 40, 46]. By doing so, the Markovian source model has a limitation of finite memory, i.e. $P[I_t|I_{t-1}, I_{t-2}]$. I_t is the transmitted quantized index at time t. This new model achieves infinite memory of a constrained form. When testing within training set, the result has shown that the signal to noise ratio of our JSC decoder for the second order hidden Markov source model is between second and third order SAMMSE decoder[40]. Although the computational complexity of our JSC decoder for the second order hidden Markov source model is similar to the standard third order SAMMSE decoder, our memory complexity is much lower and comparable to a second order SAMMSE decoder. We have also tested our approach outside the training set, as will be shown in chapter

5. However, the results are not consistent as we compared our approach to the standard SAMMSE decoder[40] with the same order using several different sources. This is a model overfitting problem, and we may need to be selective on sources where model cost needs to be measured.

As an extension, we also tried to model the source data as a Markov random field with our sequence based mean field annealing method as the joint source channel decoder. This idea was first tried in an image segmentation context by Miller et al [42]. This method relaxes the independence between pixels assumptions in the standard mean field annealing by assuming the independence is only between the rows of a image. By introducing more dependency between symbols into our source model, we could further utilize this dependency to combat the noisy channel.

1.2 Technical Background

This section provides the fundamentals to understanding the proposed methods in this thesis.

In section 1.2.1, quantization is introduced. Quantization is a lossy operation to achieve compression which maps a continuous source sequence $\underline{x}$ to a discrete source sequence $\underline{i}$ by using a look-up table. This process outputs the encoded source sequence $\underline{i}$ which could be passed through a channel coder before transmitting through noisy channel. Differential pulse code modulation, introduced in section 1.2.1.1, is a good approach for correlated sources. Trellis coded quantization, introduced in section 1.2.1.2, is a state-of-art quantization technique with low complexity but performance that may approach performance of high dimensional vector quantization.

In section 1.2.2, transmission over a noisy channel is discussed. In this thesis, we do not include explicit channel decoding in addition to the joint source channel decoding. However, the decoding techniques we propose can be used in connection with explicit channel coding when the error rate is high. In particular, inner-outer decoding could be performed. Alternatively, regular decoding and JSC decoding can be used in an iterative decoding framework [2, 20]. In our approach without explicit channel coding, after the encoded source symbol sequence $\underline{i}$ is passed through a noisy channel, the decoder receives a noisy symbol sequence $\underline{j}$, which is used in the decoder to determine the most likely transmitted encoded source symbol sequence $\hat{\underline{i}}$.

In section 1.2.3, the hidden Markov model (HMM)[49] is introduced. Markov models have been widely used to statistically model encoded source symbols. This model has been shown to be a good model for speech[1, 51] and image[17, 34], due to the residual redundancy incurred by the suboptimality of source encoders. In joint source channel decoding, Miller and Park [40] made the observation of seeing the tandem of source and channel as a hidden Markov model at the decoder. In HMM, the observable is the noisy symbol sequence $\underline{j}$, and the hidden state sequence is the encoded source symbol sequence $\underline{i}$.

In section 1.2.4, joint source channel (JSC) decoding is introduced. There are three fundamental JSC decoder structures: sequence MAP (maximum *a posteriori*), symbol MAP, and MMSE (minimum mean square error). In both sequence MAP and symbol MAP decoders, a lookup table is needed. This lookup table provides an estimation of $E[x_t|I_t = \hat{i_t}]$, which can be obtained from the training set using a centroid rule. This table gives a one to one mapping from the decoded indices to a finite set

of reconstruction values. Minimum mean squared error decoder finds the *a posteriori* probability of the decoded symbol given the received symbol sequence, and then uses a conditional mean estimator[40] to estimate the transmitted quantization levels. Thus, it effectively uses an infinite set of reconstruction values. Sequence MAP decoder and symbol MAP decoder are introduced in section 1.2.4.1. MMSE decoder is introduced in section 1.2.4.2. In this thesis, we propose novel decoders which allow bridging the gap of complexity-performance trade-off in a JSC decoder. We are able to achieve some of the performance of a high order JSC decoder without its exponential computational complexity increase and large training set requirements.

In section 1.2.5, conditional entropy-constrained encoding is introduced. Conditional entropy-constrained encoding is a lossy operation with the operational rate-distortion curve being the performance measure for the source encoder designed. We will introduce the standard greedy conditional entropy-constrained encoding[30] in section 1.2.5.1. This encoding algorithm is widely used due to its simplicity. It is also widely understood that a high order statistical model would capture more statistical dependence between symbols hence translates to better encoding performance. However, the *exponential* increase in memory storage and *large* training set size needed with model order increase prevent practical implementation using modern computers. We further applied our bridging technique to reduce both the *exponential* increase in memory storage and *large* training data set needed with model order increase. By doing so, the high order conditional entropy-constrained encoding can be implemented with significant gain in encoding performance.

In section 1.2.6, improved iterative scaling is introduced (IIS) [4]. Increasing the order of both hidden Markov and Markov model translates to improved performance for both JSC decoder and conditional entropy-constrained encoder. However, it comes with exponential complexity increase and large training set requirement to ensure good learning of the model. Hidden Markov and Markov model for encoding and decoding with order higher than two or three is typically not implementable due to these reasons. In this thesis, we propose using IIS to reduce the large training set requirement for both decoding and encoding. Our proposed algorithms used model orders as high as 13 with 10 training images for JSC decoding and 23 training images for entropy-constrained encoding applications. In order to keep the number of elements per each joint probability cell the same at this model order using direct frequency counts, we would need as many as 10 thousand training images to achieve model learning using direct implementation of JSC decoding and 20 thousand training images in conditional entropy-constrained encoding. This point will become clear in chapter 2.

1.2.1 Quantization

Quantization[53] is a lossy operation which enables discrete representation of analog sources. This is a simple process of achieving lossy data compression. It is lossy because the original analog source values cannot be reproduced at the decoder. This process involves forming a lookup table in order to map the infinite set of continuous values to a limited set of predetermined values. The mapping process is done by finding the minimum Euclidean distance between the continuous value and these predetermined values. The predetermined values are called quantization levels. Each quantization level

is represented by a binary code word. The quantization process creates new source called encoded source, with the value at each time referred to as an encoded source symbol. This new source symbol is also called the quantized symbol or the transmitted symbol. The quantization lookup table is called a code book, and the size of a code book is the number of code words used to represent the continuous source. Let N denotes the size of a code book. Thus, the bit rate of a fixed length scalar quantizer is given by $\lceil \log_2 N \rceil$. There are many ways of designing a code book and forming the binary representation for each source symbol. The input and output set of a quantizer can either be scalar or vector. If the input and output set are scalar, it is called scalar quantization. If the input and output set are vector, it is called vector quantization[21]. These quantization design techniques usually use an optimization technique to assign more levels to higher probability source regions in order to achieve less quantization error compared to a uniform quantizer with a fixed number of levels per unit volume. Moreover, each quantization level can either be represented by a fixed number of bits which is called fixed length coding or a variable number of bits which is called variable length coding. Variable length coding assigns a shorter code length to the more likely occurring source symbols which translates to more compression compared to fixed length coding. However, variable length coding is more vulnerable to noisy channels due to error propagation.

1.2.1.1 Differential Pulse Code Modulation

A more effective way of quantizing a highly correlated source symbol sequence (e.g.: speech, image, video) is by using differential encoding followed by quantizing the residue. In the setting of differential pulse code modulation (DPCM)[53], the differential

coding is merely the difference between the current unquantized source value and a prediction of this value at the current symbol time. This difference is then quantized to form residual source symbols before transmitting through the channel.

By using differential pulse code modulation, the residual source symbols have smaller dynamic range. Thus, a finer quantizer can be used to achieve less quantization error compared to a scalar quantizer at the same rate. However, error propagation could be severe in noisy channel decoding due to the fact that the decoding of current symbol depends on the decoded previous symbol sequence.

Let X_t be the unquantized source at time t , $\tilde{X}_t$ be the predicted value of X_t, Z_t be the residue of difference between X_t and $\tilde{X}_t$ at time t, $\hat{Z}_t$ be the quantized value of Z_t at time t, and $\hat{X}_t$ be the quantized value of X_t. Q is the quantizer, and P is the predictor. The output of the predictor is the prediction sequence $\{\tilde{X}_t\}$ given by $\tilde{X}_t = f(\hat{X}_{t-1}, \hat{X}_{t-2}, \ldots, \hat{X}_0)$. The basic Differential Pulse Code Modulation encoder is shown in Figure 1.1. Its matching decoder is shown in Figure 1.2.

1.2.1.2 Trellis Coded Quantization

Trellis coded quantization (TCQ) originated from Ungerboeck's set partitioning ideas from Trellis coded modulation[59]. It was developed to achieve good quantization error performance with low complexity [36]. Its quantization and noisy channel performance is excellent compared to both scalar quantization and vector quantization. The reason for its quantization performance advantage is that TCQ allows for postponing a decision of which quantization level to use at each symbol time until sequences of decisions ending at each state at final symbol time are observed. This way, the sequence

of decisions with the lowest amount of average distortion could be chosen. The reason for its decoding performance advantage in noisy channel condition is that the effect of making an error decision on trellis state will not propagate beyond $1 + log_2 N$ outputs due to the trellis structure, where N is the number of states. This makes TCQ relatively insensitive to channel noise. For these reasons, TCQ is widely used in wavelet and transform coding in noisy channel transmission frameworks.

TCQ uses a scalar code book and dynamic programming algorithm to achieve minimum squared distance between the true source sequence and quantized source sequence. Its code book is designed via an extension of the generalized Lloyd algorithm (GLA) that uses dynamic programming for nearest neighbor encoding at each algorithm iteration. At each iteration, a code book is designed for each state and a code word is chosen by nearest neighbor encoding of long sequence of source samples via dynamic programming followed by centroid updates of codebooks.

The total number of reconstruction levels for a common version of TCQ is $2^{(1+\text{bit rate})}$. The overall reconstruction levels are grouped into subsets, each of size of $2^{(\text{bit rate})}$. The way to group the elements in the subsets for a 2 bit TCQ with 4 states is shown in Figure 1.3. For each state of TCQ, only one subset can be used for quantization at each symbol time, and a code book is designed for each state. When encoding, the choice of code word also determines the choice of next state, with the state transitions limited by the structure of the TCQ. The structure of a four state TCQ with bit rate = 2 is shown in Figure 1.4.

1.2.2 Transmission Over Noisy Channel

The basic communication system considered in this thesis is shown in Figure 1.5. A simple memoryless binary symmetric channel which represents the combination of modulation, continuous channel, and demodulation blocks is used. The choice of a (discrete) binary symmetric channel in this thesis is for simplicity reasons. Our proposed techniques can also be easily extended and applied to a channel with memory. Let $\underline{X}$ be the unquantized source sequence where X_t is the unquantized source at time t. In the communication system setup used in this thesis, $\underline{X}$ is first quantized, then source encoded. The source encoded symbol sequence $\underline{I}$ is then transmitted through a noisy channel with a specified bit error rate. The noisy symbol sequence $\underline{J}$ received at the joint source channel decoder is then decoded using methods that will be described later in this thesis. Although an explicit channel coder is not used in this thesis, it can be added to our setup to further improve the decoding performance[1]. We will emphasize the performance gain of our new joint source channel decoder without the added complexity of a channel coder and decoder.

In the case of a memoryless channel, we can simplify the joint probability of the received noisy symbol sequence $P[\underline{J} = \underline{j}|\underline{I} = \underline{i}]$ in the following equation,

$$P[\underline{J} = \underline{j}|\underline{I} = \underline{i}] = \prod_{t=1}^{t=T} P[J_t = j_t|I_t = i_t]. \tag{1.1}$$

where J_t and I_t are the random variable of $\underline{J}$ and $\underline{I}$ respectively at time t. j_t and i_t are the realizations of J_t and I_t respectively.

[1] When the channel conditions are especially severe, this may be necessary.

1.2.3 Hidden Markov Model

Markov models have been widely used by researchers in speech, image, and video due to their simplicity and ability to capture statistical dependency in the case of highly correlated sources. An optimal source encoder completely removes statistical dependency between source-encoded indices to achieve maximum compression. However, an optimal source encoder often comes with high complexity and large memory. The statistical dependency, which results from sub-optimality of a source encoder, has been found to be modeled well using a Markov model[49]. Let $\underline{I}$ denote the source-encoded indices which also represent the states in an associated Markov model. Let the probability of a state sequence be $P[I_{t=1} = i_{t=1}, I_{t=2} = i_{t=2}, I_{t=3} = i_{t=3}, \ldots, I_{t=T} = i_{t=T}]$, where T is length of the sequence. In this state sequence, the states could represent encoded indices for speech waveform, row of an image, etc. This state sequence probability could be rewritten in the following equation,

$$P[\underline{I} = \underline{i}] = P[I_1 = i_1] \prod_{t=2}^{t=T} P[I_t = i_t | I_{t-1} = i_{t-1}, I_{t-2} = i_{t-2}, \ldots, I_1 = i_1]. \qquad (1.2)$$

Without further simplification, this state sequence probability requires a large amount of memory when the sequence is long. Therefore, a Markov model assumption is often used for simplification, and has been shown to be a good model for highly correlated encoded-source indices in speech[50], image[56], etc. The order of the Markov model indicates how many states of the immediate past is the current one depending on. In the case of a first order Markov model, the probability of a state sequence can be simplified

to the following,

$$P[\underline{I} = \underline{i}] = P[I_1 = i_1] \prod_{t=2}^{t=T} P[I_t = i_t | I_{t-1} = i_{t-1}]. \tag{1.3}$$

For JSC decoding, the encoded-source indices are often modeled as Markovian[15, 29, 43, 57]. These encoded-source indices are then transmitted through a memoryless noisy channel. The noisy symbol sequence at the output of the noisy channel is the input to a JSC decoder. In this context, Miller and Park[40] observed that the source and channel tandem is a hidden Markov model[50] as seen by the decoder. At the JSC decoder, the only observation in the hidden Markov model is the noisy symbol sequence $\underline{j}$, where the state sequence $\underline{i}$ (transmitted symbol sequence) is unknown (hidden) to the decoder. By using the machinery/techniques from hidden Markov modeling, we can obtain the most probable decoded symbol sequence $\hat{\underline{i}}$, the most probable decoded symbol $\hat{i}_t$, or decoded symbol probabilities $P[I_t = i_t|\underline{j}]$ at time t, given the received noisy symbol sequence $\underline{j}$.

In order to find the most probable state sequence, $\hat{\underline{i}} = \arg \max_{\underline{i}} P[\underline{I} = \underline{i}|\underline{J} = \underline{j}]$ needs to be satisfied. This sequence can be found by dynamic programing (the Viterbi algorithm)[18]. This algorithm includes initialization, recursion, termination, and backtracking[50]. Let $\delta_t[l]$ be the maximum probability metric ending in state l at time t, and $\psi_t[l]$ be the best state at the previous time through which state l is reached at time t, i.e.,such that the best metric in l at time t is achieved. Also, let N to be the number of possible decoded symbols at time t. Let us consider a first order Markov

model. Initialization and recursions are given by

$$\delta_1(l) = P[I_1 = l]P[J_1 = j_1|I_1 = l], \qquad l = 0, \ldots, N-1$$

$$\delta_t(l) = \max_{1\le k\le N}\{\delta_{t-1}(k)P[I_t = l|I_{t-1} = k]\} \cdot P[J_t = j_t|I_t = l], \quad \begin{cases} l = 0, \ldots, N-1 \\ t = 2, \ldots, T \end{cases}$$

$$\psi_t(l) = \arg\max_{1\le k\le N}\{\delta_{t-1}(k)P[I_t = l|I_{t-1} = k]\}, \quad \begin{cases} l = 0, \ldots, N-1 \\ t = 2, \ldots, T. \end{cases} \tag{1.4}$$

Termination and backtracking are given by

$$\begin{aligned} \hat{i}_T &= \arg\max_{1\le k\le N}\delta_T(k) \\ \hat{i}_t &= \psi_{t+1}(\hat{i}_{t+1}), \qquad t = T-1, \ldots, 1. \end{aligned} \tag{1.5}$$

In addition to (or instead of) finding the most probable state sequence $\underline{\hat{i}}$ given the observed sequence $\underline{\hat{j}}$, the probability of each state given the observed sequence $P[I_t = l|\underline{J} = \underline{j}]$ can also be obtained by using the well-known forward/backward (F/B) algorithm[3]. Let us consider a first order Markov model. We first rewrite the probability as follows,

$$\begin{aligned} P[I_t = l|\underline{J} = \underline{j}] &= \frac{P[I_t = l, \underline{J} = \underline{j}]}{P[\underline{J} = \underline{j}]} \\ &= \frac{P[I_t = l, \underline{J} = \underline{j}]}{\sum_{m=1}^{N} P[I_t = m, \underline{J} = \underline{j}]}. \end{aligned} \tag{1.6}$$

The numerator can be rewritten by the Markov assumption as follows,

$$P[I_t = l, \underline{J} = \underline{j}] = P[j_1, j_2, \ldots, j_t, I_t = l] P[j_{t+1}, j_{t+2}, \cdots, j_T | I_t = l]. \tag{1.7}$$

The first term is defined as the forward probabilities

$$\alpha_t[l] \equiv P[j_1, j_2, \ldots, j_t, I_t = l]. \tag{1.8}$$

The second term is defined as the backward probabilities

$$\beta_t[l] \equiv P[j_{t+1}, j_{t+2}, \ldots, j_T | I_t = l]. \tag{1.9}$$

(1.6) can then be written in terms of forward variable α and backward variable β as follows,

$$P[I_t = l | \underline{J} = \underline{j}] = \frac{\alpha_t[l]\beta_t[l]}{\sum_{m=1}^{N} \alpha_t[m]\beta_t[m]}. \tag{1.10}$$

Forward/backward probabilities can be calculated recursively as shown in [50]. The recursive equation for forward probability is as follows,

$$\begin{aligned}
\alpha_1(l) &= P[I_1 = l]P[J_1 = j_1|I_1 = l], \qquad l = 1, \ldots, N \\
\alpha_t(l) &= \sum_{k=1}^{N} P[j_1, j_2, \ldots, j_t, I_{t-1} = k, I_t = l] \\
&= \sum_{k=1}^{N} P[j_1, j_2, \ldots, j_{t-1}, I_{t-1} = k]P[J_t = j_t, I_t = l|I_{t-1} = k] \\
&= \sum_{k=1}^{N} \alpha_{t-1}(k)P[J_t = j_t|I_t = l]P[I_t = l|I_{t-1} = k] \\
&= \{\sum_{k=1}^{N} \alpha_{t-1}(k)P[I_t = l|I_{t-1} = k]\} \cdot P[J_t = j_t|I_t = l]. \qquad (1.11) \\
&\quad l = 1, \ldots, N, \qquad t = 2, \ldots, T
\end{aligned}$$

The recursive equation for backward probability is as follows,

$$\begin{aligned}
\beta_T(l) &= 1, \qquad l = 1, \ldots, N \\
\beta_t(l) &= \sum_{k=1}^{N} P[j_{t+1}, j_{t+2}, \ldots, j_T, I_{t+1} = k|I_t = l] \\
&= \sum_{k=1}^{N} P[I_{t+1} = k|I_t = l]P[j_{t+1}, j_{t+2}, \ldots, j_T|I_{t+1} = k] \\
&= \sum_{k=1}^{N} P[I_{t+1} = k|I_t = l]P[J_{t+1} = j_{t+1}|I_{t+1} = k]P[j_{t+2}, \ldots, j_T|I_{t+1} = k] \\
&= \sum_{k=1}^{N} P[I_{t+1} = k|I_t = l]P[J_{t+1} = j_{t+1}|I_{t+1} = k]\beta_{t+1}(k). \qquad (1.12) \\
&\quad l = 1, \ldots, N, \qquad t = T-1, \ldots, 1
\end{aligned}$$

Once the α and β are calculated, the $P[I_t = l|\underline{J} = \underline{j}]$ can be calculated using (1.10). It is apparent that directly implementing these recursive equations would result in numerical underflow. Therefore, a renormalization step for both α and β must be done at each iteration to avoid this problem [50].

1.2.4 Joint Source-Channel Decoding

There are two types of decoder when the source-channel tandem in a communication system is modeled as a hidden Markov model[50]. The first type is a maximum a posterior probability (MAP) decoder[45, 48]. The second type is a minimum mean square error (MMSE) decoder[40]. In both types of decoders, the transmitted symbol sequence $\underline{I}$ is considered as hidden states (not known at the decoder). The only observable at the decoder is the noisy symbol sequence $\underline{J}$ after the channel which is generated from hidden state sequence $\underline{I}$ according to the channel transition probability, i.e. $\{P[J_t = j_t|I_t = i_t]\}$.

1.2.4.1 Maximum *A Posteriori* Probability Decoder

A MAP decoder which maximizes the hidden state sequence probability $P[\underline{I} = \underline{i}|\underline{J} = \underline{j}]$ is called a sequence MAP decoder[48]. A MAP decoder which maximizes the hidden state probability $P[I_t = i_t|\underline{J} = \underline{j}]$ at time t is called a symbol MAP decoder[45]. Although a sequence MAP decoder is somewhat computationally less complex compared to a symbol MAP decoder, a symbol MAP decoder was found to outperform a sequence MAP decoder by as much as 2dB in signal to noise ratio when the channel error rate is high[40]. At low channel error rates, both decoders perform about the same.

The sequence MAP decoder finds the decoded symbol sequence which maximizes $P[\underline{I} = \underline{i}|\underline{J} = \underline{j}]$ by using the Viterbi algorithm on a trellis structure. The metric associated with each transition is a function of $P[I_t = l|I_{t-1} = m]$ and $P[j_t|I_t = l]$. The best path is found by the algorithm introduced in (1.4). The sequence MAP decoder has the advantage of being able to constrain the decoded symbol sequence format to match the encoder. This may be important in cases when invalid sequence format is not tolerable.

The symbol MAP decoder finds the decoded symbol at time t which maximizes $P[I_t = l|\underline{J} = \underline{j}]$. These probabilities are computed by the forward/backward algorithm[3] introduced in (1.10). This algorithm finds the symbol *a posteriori* probability $P[I_t = l|\underline{J} = \underline{j}]$ $\forall t$, and chooses the decoded symbol to be the one with maximum symbol *a posteriori* probability.

In both sequence MAP and symbol MAP decoders, a lookup table is needed. The lookup table provides an estimation of $E[x_t|I_t = \hat{i_t}]$, which can be obtained from the training set using a centroid rule. This table gives a one to one mapping from decoded indices to a finite set of reconstruction values.

1.2.4.2 Minimum Mean Squared Error Decoder

As opposed to a MAP decoder which makes hard decisions, a MMSE decoder[40] is a soft-input-soft-output (SISO) decoder. This is because the MMSE decoder first calculates the symbol *a posteriori* probability $P[I_t = l|\underline{J} = \underline{j}]$ $\forall t$ then uses the conditional mean estimator to estimate the quantization levels transmitted. Instead of using a look-up table which has a finite set of reconstructed values as in the MAP decoder, an MMSE decoder effectively uses an infinite set of reconstructed values. This leads to better

performance. The conditional mean estimator developed by Miller and Park[40] is an approximation to the optimal decoder minimizing the mean squared error. It takes the statistical average of quantization levels weighted by the symbol *a posteriori* probability at each symbol time.

Let $\underline{y} = (y_1, y_2, \ldots, y_T)$, $y_t \in \mathcal{R}$ to be the reconstructed values for the source sequence $\underline{x} = (x_1, x_2, \ldots, x_T)$, $x_t \in \mathcal{R}$ before quantization. The overall distortion which is caused by quantization error and decoder error is given by $D(\underline{x}, \underline{y(j)}) = \sum_{t=1}^{t=T} d(x_t, y_t(\underline{j}))$, where $d(x_t, y_t(\underline{j})) = (x_t - y_t(\underline{j}))^2$. The objective here is trying to minimize the expected distortion over $\underline{y}$ given the received noisy symbol sequence $\underline{j}$[2]. This equation is given by

$$E[D(\underline{x}, \underline{y})|\underline{J} = \underline{j}] = \frac{\sum_{\underline{i}} \Big(\sum_{t=1}^{t=T} E[d(x_t, y_t)|\underline{I} = \underline{i}] \Big) P[\underline{J} = \underline{j}|\underline{I} = \underline{i}] P[\underline{I} = \underline{i}]}{\sum_{\underline{i}} P[\underline{J} = \underline{j}|\underline{I} = \underline{i}] P[\underline{I} = \underline{i}]}. \tag{1.13}$$

The authors in [40] have shown that directly working with this equation is not practical. Therefore, an approximation of $E[d(x_t, y_t)|\underline{I} = \underline{i}] \approx E[d(x_t, y_t)|I_t = i_t]$ is needed. The decoder using this approximation is called a sequence-based approximate MMSE (SAMMSE) decoder. By using this approximation, (1.13) can be further reduced to minimizing

$$\sum_{t=1}^{t=T} \sum_{l=1}^{l=L} \|E[x_t|I_t = l] - y_t\|^2 P[I_t = l|\underline{J} = \underline{j}] \tag{1.14}$$

[2]For simplicity purpose, we may write y_t in place of $y_t(\underline{j})$ for the rest of this thesis. It is clear that our reconstructed value y_t is always a function of the decoder input $\underline{j}$.

,where

$$P[i_t = l|\underline{J} = \underline{j}] = \frac{\sum\limits_{\underline{i}:i_t=l} p[\underline{J} = \underline{j}|\underline{I} = \underline{i}]P[\underline{I} = \underline{i}]}{\sum\limits_{\underline{i}} P[\underline{J} = \underline{j}|\underline{I} = \underline{i}]P[\underline{I} = \underline{i}]}. \tag{1.15}$$

This minimization leads to choosing the reconstructed values y_t according the centroid rule

$$y_t = \sum_{l=1}^{l=L} E[x_t|I_t = l]P[I_t = l|\underline{J} = \underline{j}], \qquad \forall t. \tag{1.16}$$

Note that $P[I_t = l|\underline{J} = \underline{j}]$ is obtained by using the forward/backward algorithm introduced in (1.10), and $E[x_t|I_t = l]$ is simply a lookup table obtained from a large training set. The reader is refereed to [40] for a detailed development on this subject.

1.2.5 Conditional Entropy-Constrained Encoding

Shannon has proved that a source encoder could be designed to achieve its entropy rate without any loss of information[16, 55]. Many researchers have tried to find such source encoder but it usually comes with high complexity. An alternative to this is the lossy source encoder, where the source bit rate could be lower than its entropy rate with lower complexity for the price of distortion. In lossy source encoding, the operational rate-distortion curve[12] is a performance measure for the source encoder designed. Entropy-constrained encoding was proposed to provide a good operational rate-distortion curve by the means of variable length coding[16, 36]. In image encoding, a conditional entropy-constrained encoding is sometimes used to capture statistical dependency within and between source blocks after linear frequency transformation (e.g.

DCT)[7, 10, 28]. This statistical dependency is specified by $P[i_{m,n}|i_{m,n-1}, i_{m-1,n}]$ assuming a second order Markov model, where $i_{m,n}$ is the quantization index at row m and column n. $i_{m,n-1}$ and $i_{m-1,n}$ are to the left and top of $i_{m,n}$ respectively. In the following sections, we are using images as our source to encode.

The objective of a conditional entropy-constrained encoder is to minimize the Lagrangian cost function $J \equiv D + \lambda \cdot R$ over choice of quantization indices, where D is the distortion, and R is the bit rate. For better understanding of the following sections, this overall Lagrangian cost is rewritten as the sum of Lagrangian cost for each row, i.e.

$$J = (D_1(\underline{i_1}) + \lambda R_1(\underline{i_1})) + \sum_{m=2}^{m=M} (D_m(\underline{i_m}) + \lambda R_m(\underline{i_m}, \underline{i_{m-1}})) \tag{1.17}$$

where,

$$\begin{aligned} D_m(\underline{i_m}) &= \sum_{n=1}^{n=N} d(s_{m,n}, q(i_{m,n})) \\ R_1(\underline{i_1}) &= l(i_{1,1}) + \sum_{n=2}^{n=N} l(i_{1,n}; i_{1,n-1}) \\ R_m(\underline{i_m}, \underline{i_{m-1}}) &= l(i_{m,1}; i_{m-1,1}) + \sum_{n=2}^{n=N} l(i_{m,n}; i_{m-1,n}, i_{m,n-1}) \qquad M \geq m \geq 2. \end{aligned}$$

$s_{m,n}$ is the source value at row m and column n before quantization. The source value $s_{m,n}$ can either represent pixel value in spatial domain or transformed value in frequency domain. The detailed setup of our transformed coding for images will be introduced in the next chapter. $q(i_{m,n})$ is the quantization value representing quantization index $i_{m,n}$. $d(s_{m,n}, q(i_{m,n}))$ is the distance function, which we choose as squared distance.

$l(i_{m,n}; i_{m-1,n}, i_{m,n-1})$ is the bit length describing quantization index $i_{m,n}$ as a function of $i_{m-1,n}$ and $i_{m,n-1}$.

1.2.5.1 Standard Greedy Conditional Entropy-Constrained Encoding

Exhaustive search over all possible $\prod_{m=1}^{m=M} (L_m)^N$ image encodings provides a globally optimum solution. However, the search space is too large even for the smallest practical image size. A simple alternative is the standard greedy approach[30]. This approach does not guarantee even a locally optimum solution. However, the simplicity of this approach has earned its usage in many applications.

The standard greedy approach optimizes its solution row-by-row given the previously encoded row. The quantization indices for each row are chosen via Viterbi algorithm with the Lagrangian cost on each transition. Once the quantization indices are chosen for the current row, they are being used in choosing the quantization levels for the next row. The process is repeated until the whole image has been passed. The pseudo code for the standard greedy approach is as follows:

1. Use Viterbi algorithm to minimize $D_1(\underline{i_1}) + \lambda R_1(\underline{i_1})$. Let $i_1^{(0)}$ denote the solution.

2. For $m = 2$ to M

   Use Viterbi algorithm to minimize $D_m(\underline{i_m}) + \lambda R_m(\underline{i_m}, \underline{i_{m-1}^{(0)}})$ over $\underline{i_m}$. Let $\underline{i_m}^{(0)}$ denote the solution.

   End

1.2.6 Improved Iterative Scaling

It has been shown that increasing order of Markov model can increase performance in both JSC decoding[44] and conditional entropy constrained encoding[6]. However, directly increasing model order results in increasing both memory storage and computational complexity exponentially. The training data support needed for learning the statistical model also increases substantially. Due to these reasons[3], a Markov model with order higher than two or three with reasonable number of states is usually not implementable using modern computers. In this thesis, we propose a bridging technique to improve performance using higher order models via a special version of the maximum entropy (ME) algorithm[41] - improved iterative scaling(IIS)[4]. By using IIS[4], a higher order conditioning model (with special structure) with low memory requirement and computational complexity are achieved without needing a large training data support for statistical model learning.

The problem of finding a higher order probability model using a maximum entropy approach with lower order probability constraints was first proposed by Cheeseman[8], and since then has been addressed by several researchers[22, 25, 41] in the field of statistical classification. This maximum entropy model is powerful due to not having to assume conditional independence such as in a naive Bayes model. In this thesis, we used second order probabilities as our lower order constraints which are measured from frequency counts using training data sets. Let us consider the case of image coding and decoding at this point. Let C denotes the quantization index of current pixel, where

[3]These reasons will be explained in detail in chapter 2

the conditioning context[4] in neighboring pixels is given by a feature vector $\underline{f}$. Let K denote the number of possible quantization values for the current pixel and N denote the feature vector dimension. Lower order probability constraints on pairs of quantization values for neighboring pixels are enforced using Lagrange mutilpliers $\gamma(f_i, C = c)$, which need to be learned. The higher order conditional probability consistent with all lower order constraints (all pairwise constraints $P[f_i, C = c]$) derived from the context is given as an exponential model,

$$P[C = c|\underline{f}] = \frac{e^{(\sum_{i=1}^{N} \gamma(f_i, C=c))}}{\sum_{c'=1}^{K} e^{(\sum_{i=1}^{N} \gamma(f_i, C=c'))}} \qquad c = 1, \ldots, K. \tag{1.18}$$

In order to find the maximum entropy joint PMF while satisfying the lower order constraints using Lagrange multipliers $\gamma(f_i, C = c)$ by using uniform assumption ($\underline{f}$ in training set all assumed equally likely), it can be shown that the ME problem becomes equivalent to maximizing the conditional log likelihood $L = \frac{1}{N_t} \sum_{(\underline{f},c) \in \text{training set}} \log P[C = c|\underline{f}]$ [5], where N_t is the number of all possible feature vectors $\underline{f}$. This is the objective of IIS[4]. IIS[4] provides an iterative process to maximize the likelihood by updating the Lagrange Multipliers $\gamma(f_i, C = c)$ through iterations. Each iteration is indexed by superscript (k). The iteration steps are,

1. Randomly initialize the Lagrange Multipliers $\underline{\gamma}$.

[4] also quantized indices

[5] with $P[C = c|\underline{f}]$ given by (1.18)

2. Calculate the increments $\underline{\Delta\gamma^{(k)}}$ for Lagrange multipliers which guarantees non-decreasing likelihood cost.

3. Add the increments to the Lagrange multipliers, $\underline{\gamma^{(k+1)}} = \underline{\gamma^{(k)}} + \underline{\Delta\gamma^{(k)}}$

   $k = k + 1$.

4. iterate 2 and 3 until convergence.

$\underline{\Delta\gamma}$ is obtained by specifying an auxiliary function $A(\underline{\Delta\gamma}, \underline{\gamma})$, which has a unique maximum $\max_{\underline{\Delta\gamma}} A(\underline{\Delta\gamma}, \underline{\gamma}) \geq 0$, and $L(\underline{\gamma} + \underline{\Delta\gamma}) - L(\underline{\gamma}) \geq A(\underline{\Delta\gamma}, \underline{\gamma})$. The auxiliary function for our ME problem is given by,

$$A(\underline{\Delta\gamma}, \underline{\gamma}) = \sum_{j=1}^{N_c} \sum_{c=1}^{K} \sum_{l=1}^{|A_j|} \Delta\gamma(C = c, f_j = l) \frac{N(C = c, f_j = l)}{T} + 1 - \sum_{j=1}^{N_c} \sum_{c=1}^{K} \sum_{l=1}^{|A_j|} \frac{1}{LT} \sum_{\underline{f}|f_j=l} P[C = c|\underline{f}] e^{L\Delta\gamma(C=c, f_j=l)}, \quad (1.19)$$

where L is the number of features, $N(\cdot)$ is the number of occurrence in the training data set, and T is the training data set size. After taking the partial derivative over $\underline{\Delta\gamma}$ to maximize $A(\underline{\Delta\gamma}, \underline{\gamma})$ by setting it to zero, the increments for Lagrange multipliers for each iteration can be solved as,

$$\Delta\gamma^{(k+1)}(C = c, f_j = l) = \frac{1}{L} \log \frac{N(C = c, f_j = l)/T}{\frac{1}{T} \sum_{\underline{f}|f_j=l} P[C = c|\underline{f}]}. \quad (1.20)$$

1.3 Thesis Contributions

In our first JSC decoding algorithm, we propose using IIS[4] to achieve higher order conditioning using both clean (I) and noisy symbol (J) contexts. The advantages of this algorithm are twofold. One, the support training data size needed to learn the high order statistical model is greatly reduced compared to directly learning the high order conditional probabilities via frequency counts. These conditional probabilities for learning the model directly by using frequency counts are given by $\frac{N(\underline{f},c)}{N(\underline{f})}$. In our experiment, we used 10 training images for our higher order JSC decoding with IIS. To keep the same number of training data per each probability cell in a statistical model, learning the model directly by using frequency counts would require around 10 thousand training images. Therefore, directly learning the higher order statistical model is not practical. However, since our proposed algorithm learns the model based on both clean and noisy symbols from training set and it only encodes lower order constraints, it provides an intermediate decoding performance between orders of learning the model directly by using frequency counts on either only clean symbols (computationally intractable - computational complexity increases exponentially with model order increase) or both clean and noisy symbols given the non-practical condition that enough training data is provided. The detailed setup of how we can use both clean and noisy symbol context to reduce the computational complexity of high order Markov model in JSC decoding is explained in chapter 2. Our algorithm has been verified experimentally to produce better decoding results[6] in signal to noise ratio than the standard SAMMSE

[6]The SNR of our algorithm is about 1dB better.

decoder with comparable computational complexity[7]. Our method provides a bridge between existing performance complexity trade-off choices for JSC decoding. Previously, one could only choose to vary the order of the Markov model. However, the complexity grows exponentially with increasing order. Our second advantage, as already mentioned, is that we are able to keep the computational complexity of our modified higher order (as high as 13th order) JSC decoder comparable to the first order SAMMSE decoder. The memory complexity of our proposed algorithm grows linearly with model order increase, while the memory complexity of a directly learned model grows exponentially with model order increase.

This idea of using IIS to reduce training set size needed to provide higher order conditioning is also applied to the conditional entropy constrained encoding algorithm. By using high order (as high as 13th order) conditional probability model obtained through IIS learning (using 23 training images), we are able to improve the encoding SNR performance by as much as 1dB compared to a comparable standard greedy conditional entropy constrained approach in low bit rate. This higher order conditioning is not practical in direct implementation due to the number of training set data required for learning the model for the same reason as mentioned in the high order JSC decoding case. This idea also provides a bridge in performance complexity trade-off for entropy constrained encoding.

We also introduce another bridging technique for performance complexity trade-off in JSC decoding. This is a two stage approach based on first applying the low order SAMMSE decoding, then least squares (LS)[24] filtering to further improve the

[7]First order hidden Markov model is used in our experiments

low order SAMMSE decoding performance. This algorithm first passes a set of training symbols through a noisy channel and a standard low order SAMMSE decoder, then uses the SAMMSE decoded training symbols to learn the LS filter coefficients. These LS filter coefficients are then used on the test symbol sequence after (first stage) SAMMSE decoding. The computational complexity of a SAMMSE decoder is $O(N^{K+1}T)$. The computational complexity of a LS filter is $P(T-P)$. With only small number of LS filter taps P, we justified the performance gain in our experiments with the overall computational complexity increase by the computational complexity of an LS filter. This increase is much less than increasing the order of the SAMMSE decoder by one. This also provides a performance and complexity trade-off that cannot be practically achieved by previously proposed methods.

Our next bridging technique for performance complexity trade-off is by modeling the source as a hidden Markov model, while a Markov model is commonly used in JSC decoding. By using our hidden Markov source model, we are able to achieve infinite memory compared to the finite memory in a Markov source model (as will be shown in chapter 5). In this work, we also show a way to iteratively update our hidden Markov model parameters with increasing likelihood, hence reduces entropy. This also translates to higher lossless compression and better decoder performance (SNR) compared to a standard SAMMSE decoder with similar memory complexity (same order), although our computational complexity is higher. Our decoding performance will be shown to provide intermediate performance between the current order of standard SAMMSE decoder with comparable memory complexity and the next order higher when tested within training set. We found that the the decoding performance is not consistent when testing outside

the training set due to model overfitting problem. Thus, we need to be selective on sources given the model cost provided. Even though the decoding performance is not consistent when testing outside training set, the merit of our mathematical developments and possible contributions/extensions to other applications should not be overlooked.

Last, as an extension, we also tried to model the image source data as Markov random field with our sequence based mean field annealing method as the JSC decoder. This idea was first tried in a image segmentation context by Miller et al.[42]. This method relaxes the independence between pixels assumption in the standard mean field annealing method[26, 38] by assuming the independence is only between the rows of the image. By introducing more dependency between symbols into our source model, we could further utilize this dependence to combat the noisy channel.

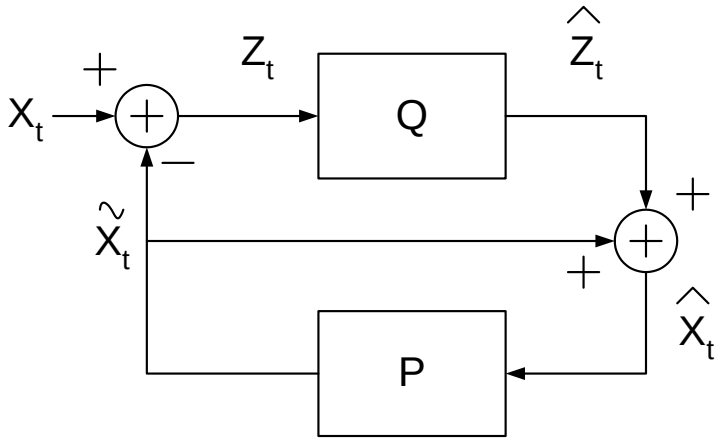

Fig. 1.1. DPCM encoder

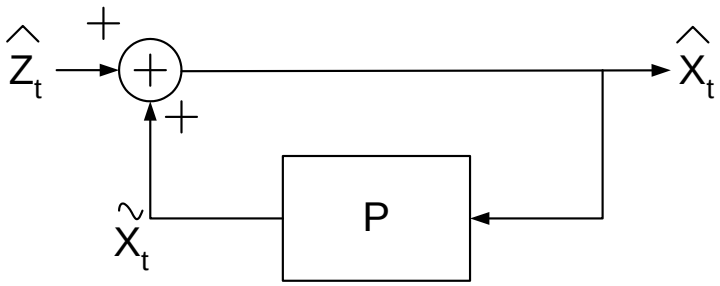

Fig. 1.2. DPCM decoder

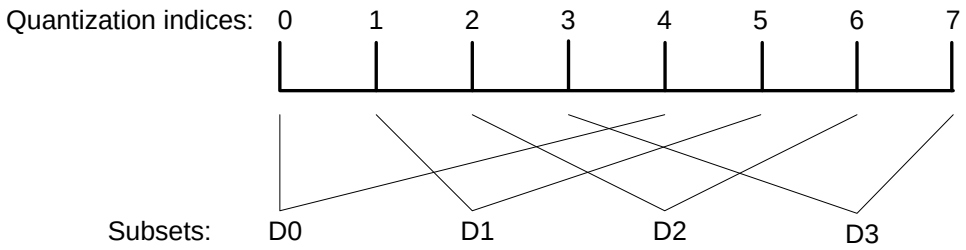

Fig. 1.3. set partitioning used in a 2 bit 4 state TCQ, D_k are subsets, 0-7 are quantization indices

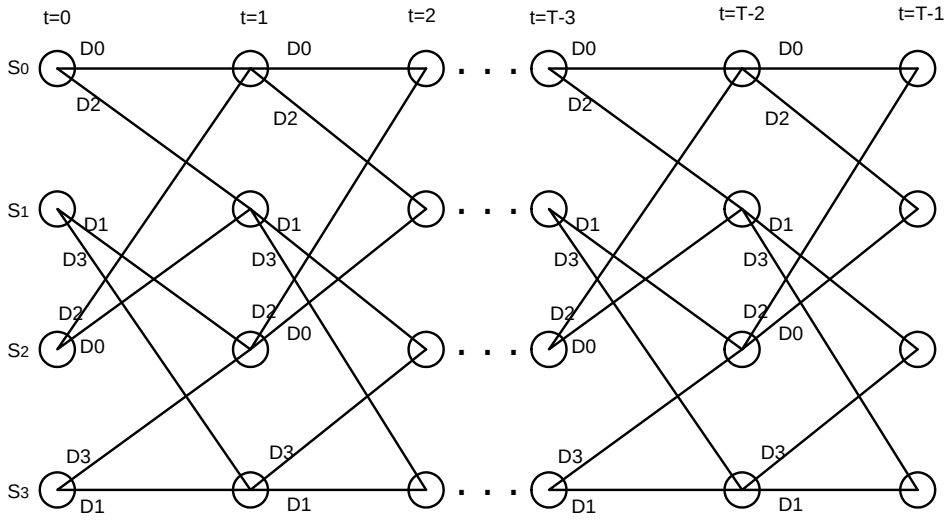

Fig. 1.4. The structure of a 4 state TCQ with bit rate = 2, S_k are states, D_m are subsets

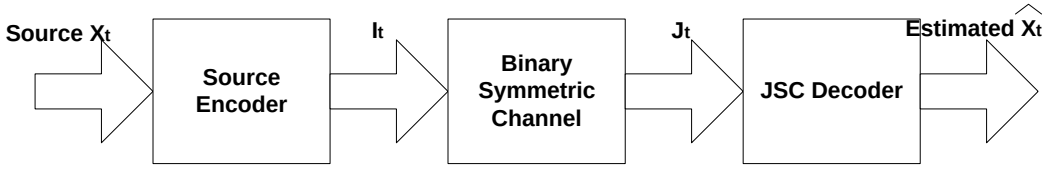

Fig. 1.5. basic communication system considered in this thesis

Chapter 2

High Order JSC Image Decoding with Reduced Complexity via Improved Iterative Scaling

2.1 Introduction

In recent years, several researchers[14, 31, 32, 46, 61] have attacked the problems of noisy channel image decoding based on the use of Markov source modeling[49]. There are three fundamental types in this framework of JSC image decoding. The first type uses the Viterbi algorithm which makes a sequence of hard decisions conditioning on the predetermined choice (from the row above) above and to the left of the current choice of decoded symbol[7, 32]. The second type uses the forward/backward algorithm to achieve soft decision using information from both causal and anticausal subsets of the observed data[31, 46]. This type of method also uses information from neighboring rows in order to obtain soft decision on the current row. Although the second type has been shown to outperform the first type, it is also computationally more complex. Both type one and two have exponential computational complexity increase with Markov model order increase. Therefore, a JSC image decoder using these two types are typically not implementable with model order greater than 2 or 3. The third type is a greedy approach[14, 30]. This type uses the hard decision made using the decoded symbol above and to the left of current symbol. It makes an immediate decision on the current symbol without the objective of minimizing the cost for the whole row or image. The computational complexity

for this greedy approach does not grow with Markov model order increase. However, this approach's decoding performance is much worse compared to the first two types. Our proposed method is the fourth type. In this chapter, we propose increasing Markov model order by using *noisy symbol neighbors* as part of conditioning context instead of increasing *hidden states*. We refer to the hidden states/encoded source symbols as *clean* symbols contrary to the received noisy symbols at the decoder. We will show that our computational complexity only increases slightly with model order increase and yet still achieves better decoding performance compared to a standard SAMMSE decoder[40] (second type) of similar computational complexity.

Besides the exponential complexity increases with Markov model order increases, all types mentioned above suffer from needing a large training data set to ensure good learning of their high order Markov source model. This problem was first addressed by Rissanen[52], and the term *model cost* was coined. The author in [52] showed that by increasing the order of conditioning without having more training data, the modeling performance will only increase with order increase up to a point. Then, the performance starts declining with conditioning order increase. This problem is important due to the fact that the order of conditioning does not have to be high (typically 3 or above) before the size of training data set needed becomes *substantial*. Wu et al.[60] proposed a method to address this problem within the context of conditional entropy coding. Their method is a Bayesian-type technique which uses lower order constraints. To our knowledge, we have not see anyone address this issue in the JSC decoding context. We only have seen researchers testing their proposed JSC decoders with order less than 3. Our proposed method in this chapter solves this problem by using the improved iterative

scaling algorithm[4] to reduce the training set data needed to ensure good learning of our high order model.

In section 2.2.2, the improved iterative scaling algorithm[4] (IIS) is introduced. IIS algorithm is used to reduce the size of the training set needed to ensure good learning of a high order hidden Markov model[49]. This translates to keeping the number of elements per each cell in the higher order joint PMF the same as in a lower order joint PMF. By using IIS[4], our algorithm achieves a high order, but structurally-constrained (parameterized) PMF, consistent with a large group of constraints on our model's high order PMF. Our model's data set requirements are determined by *low* order (pairwise cells), not *high* order ones. However, the computational complexity of our HMM would *still* grow exponentially with the order if we directly model high order conditional state probabilities. In section 2.2.3, we propose adding neighboring *noisy* symbols to the conditioning context to achieve a higher order hidden Markov model instead of directly increasing the order in the standard way. We retain low complexity by using, as much of the conditioning context, not neighboring (hidden) states, but rather neighboring received (noisy) observations. This will be explained shortly. This method would be inferior compared to directly increasing the order of a hidden Markov model by using only encoded source (clean) symbols/states. However, such an approach would have enormous complexity. We have verified that our SNR decoding performance is about 1dB better compared to a standard SAMMSE decoder[40] with similar computational complexity. Thus, our proposed decoder provides a bridge for a complexity-performance tradeoff between lower and higher order implementation of a standard SAMMSE decoder[40].

2.2 Formulation

In this section, the detailed formulation is shown. We first describe how we implement transform coding of images. We then show how the improved iterative scaling algorithm[4] could be used to significantly reduce the training set support size needed to ensure good learning of the model. Also, by using IIS[4], the memory complexity of our JSC decoder only increases linearly with model order increase compared to the exponential increase in the standard[40] approach. Last, we show how to reduce computational complexity (comparable to a first order hidden Markov model) by using both clean and noisy symbol conditioning for a high order hidden Markov model.

2.2.1 Transform Image Coding

For linear transformation of images, we first divide the image into $8X8$ blocks. Each $8X8$ block is then transformed using discrete cosine transform (DCT)[53] linear transformation. The transformed sources are formed by grouping the frequency elements from each block with the same spatial frequency, i.e. the same spatial location in each $8X8$ block. The elements in each frequency source are collected left-to-right and row-by-row from each block. Thus, the number of elements in each block, 64, is the number of frequency sources. In a $512X512$ image, there are 4096 elements in each frequency source. The sources are then sorted from highest to lowest variance. This sorting process is usually a zigzag scan in each block. The number of quantization levels for each frequency source is determined by an optimal bit allocation algorithm[21] in order to achieve a minimum distortion with the desired overall rate. Let $s_{m,n}$ denote

the transformed source value for the n_{th} element in source m. Figure 2.1 shows the detailed configuration for this setup. A TCQ[36] quantizer is then applied to each source to produce quantization indices (or encoded source symbols). Its quantization index is produced by $i_{m,n} = q_m[s_{m,n}]$, where q_m is the quantizer for the m_{th} source. Note that the quantizer for each source is in general different in quantization levels and rate.

2.2.2 Improved Iterative Scaling Algorithm

In this chapter, we used second order probabilities as our lower order constraints which are measured from frequency counts using training data sets. Let us consider the case of image coding and decoding at this point. Let $i_{m,n}$ denote the quantization index of current transformed value at m_{th} source and n_{th} element. Let $\mathbf{j}$ denote the entire noisy image ($\{j_{m,n}\} \quad \forall m, \forall n$) received at the decoder after a noisy channel. In order to reduce the memory complexity by using conditional independence, we are assuming $P[I_{m,n}|I_{m,n-1}, \mathbf{j}] = P[I_{m,n}|I_{m,n-1}, \mathbf{j_{m,n}}]$, where $\mathbf{j_{m,n}}$ is a subset of $\mathbf{j}$. This subset is the *effective noisy neighborhood* of $I_{m,n}$. The effective noisy neighborhoods are determined empirically for each source, and its detailed configuration will be shown in the experimental section. By using IIS[4], we are learning the maximum entropy *a posteriori* probability $P_{ME}[I_{m,n} = l|I_{m,n-1} = k, \mathbf{j_{m,n}}]$ from a large group of pairwise constraints ($P[I_{m,n} = l, I_{m,n-1} = k], \{P[I_{m,n} = l, j_{q,r}]\} \quad j_{q,r} \in \mathbf{j_{m,n}}$). All lower order probability constraints are enforced using Lagrange multipliers ($\gamma(I_{m,n} = l, I_{m,n-1} = k), \{\gamma(I_{m,n} = l, j_{q,r})\} \quad j_{q,r} \in \mathbf{j_{m,n}}$), which need to be learned. Let N_m denote the number of possible quantization values for the current pixel. The higher order conditional probability consistent with all lower order constraints derived from the context

and maximizing entropy is given as an exponential model[41],

$$P_{ME}[I_{m,n} = l | I_{m,n-1} = k, \mathbf{j_{m,n}}] \tag{2.1}$$
$$= \frac{e^{\gamma(I_{m,n}=l, I_{m,n-1}=k) + \left(\sum_{j_{q,r} \in \mathbf{j_{m,n}}} \gamma(I_{m,n}=l, J_{q,r}=j_{q,r})\right)}}{\sum_{l'=1}^{N_m} e^{\gamma(I_{m,n}=l', I_{m,n-1}=k) + \left(\sum_{j_{q,r} \in \mathbf{j_{m,n}}} \gamma(I_{m,n}=l', J_{q,r}=j_{q,r})\right)}} \qquad l = 1, \ldots, N_m.$$

The detailed formulation of the IIS[4] algorithm for learning these Lagrange multipliers is shown in chapter 1. Note that we need to run IIS[4] algorithm for each frequency source because each frequency source has different conditioning context and quantizer.

By using IIS[4] algorithm, we only need enough training data to ensure good learning of the second order joint probabilities by using frequency counts $N(I_{m,n} = l, I_{m,n-1} = k), \{N(I_{m,n} = l, J_{q,r} = j_{q,r})\}$ $j_{q,r} \in \mathbf{j_{m,n}}$. By contrast, in directly learning unconstrained high order model[40], we would need enough training data to ensure good learning of the high order joint probabilities by using frequency counts $N(I_{m,n} = l, I_{m,n-1} = k, \mathbf{j_{m,n}})$.

2.2.3 High Order JSC Decoding with Noisy Symbol Context

In a standard SAMMSE decoder[40], the computational complexity increases exponentially with increasing order. Thus, a standard SAMMSE decoder[40] with order greater than two or three is typically non-implementable. In this section, we propose using both clean and noisy symbols conditioning context in our high order hidden Markov model to reduce computational complexity. We will show that adding noisy symbol

neighbors to the conditioning context only incurs a slight computational complexity increase. Let us consider the image decoding case, where the decoding is done on one transformed source at a time. Let T denote the sequence length of each source. We assume a memoryless channel, which allows us to simplify the joint probability of the received noisy symbol sequence $P[\underline{J_m} = \underline{j_m} | \underline{I_m} = \underline{i_m}]$ in the following equation,

$$P[\underline{J_m} = \underline{j_m} | \underline{I_m} = \underline{i_m}] = \prod_{n=1}^{n=T} P[J_{m,n} = j_{m,n} | I_{m,n} = i_{m,n}]. \tag{2.2}$$

The *a posteriori* probability in our proposed decoder is given as follows,

$$\begin{aligned} P[I_{m,n} = l | \mathbf{j}] &= \frac{P[I_{m,n} = l, \mathbf{j}]}{P[\mathbf{j}]} \\ &= \frac{P[I_{m,n} = l, \mathbf{j}]}{\sum_{k=1}^{N_m} P[I_{m,n} = k, \mathbf{j}]}. \end{aligned} \tag{2.3}$$

The numerator can be rewritten by the Markov assumption as follows,

$$\begin{aligned} & P[I_{m,n} = l, \underline{J_m} = \underline{j_m}, \mathbf{j}^{(-\underline{j_m})}] \\ & = P[j_{m,1}, j_{m,2}, \ldots, j_{m,n}, I_{m,n} = l, \mathbf{j}^{(-\underline{j_m})}] \\ & \cdot P[j_{m,n+1}, j_{m,n+2}, \cdots, j_{m,T} | I_{m,n} = l, \mathbf{j}^{(-\underline{j_m})}], \end{aligned} \tag{2.4}$$

where $\mathbf{j}^{(-\underline{j_m})}$ is the noisy image $\mathbf{j}$ excluding the m^{th} row $\underline{j_m}$. The first term is defined as the forward probabilities

$$\alpha_{m,n}[l] \equiv P[j_{m,1}, j_{m,2}, \ldots, j_{m,n}, I_{m,n} = l, \mathbf{j}^{(-\underline{j_m})}]. \tag{2.5}$$

The second term is defined as the backward probabilities

$$\beta_{m,n}[l] \equiv P[j_{m,n+1}, j_{m,n+2}, \ldots, j_{m,T} | I_{m,n} = l, \mathbf{j}^{(-\underline{j_m})}], \tag{2.6}$$

(2.3) can then be written in terms of forward variable α and backward variable β as follows,

$$P[I_{m,n} = l|\mathbf{j}] = \frac{\alpha_{m,n}[l]\beta_{m,n}[l]}{\sum_{k=1}^{N_m} \alpha_{m,n}[k]\beta_{m,n}[k]}. \tag{2.7}$$

Forward/backward probabilities can be calculated recursively as shown in [49].

Let us consider the forward probability at this point. By using the conditional independence assumption ($I_{m,n}$ conditioning on $\mathbf{j}$ is equivalent to $I_{m,n}$ conditioning on $\mathbf{j_{m,n}}$) from the previous section, we can write $P[I_{m,n} = l|I_{m,n-1} = k, \mathbf{j}] = P[I_{m,n} = l|I_{m,n-1} = k, \mathbf{j_{m,n}}]$. This assumption is used to further simplify our forward recursive

equation. The recursive equation for forward probability is as follows,

$$
\begin{aligned}
\alpha_{m,1}(l) &= P[I_{m,1}=l|\mathbf{j_{m,1}}]P[J_{m,1}=j_{m,1}|I_{m,1}=l], \qquad l=1,\ldots,N_m \\
\alpha_{m,n}(l) &= \sum_{k=1}^{N_m} P[j_{m,1},j_{m,2},\ldots,j_{m,n},I_{m,n-1}=k,I_{m,n}=l,\mathbf{j}^{(-\underline{j_m})}] \\
&= \sum_{k=1}^{N_m} P[j_{m,1},j_{m,2},\ldots,j_{m,n-1},I_{m,n-1}=k,\mathbf{j}^{(-\underline{j_m})}] \\
&\quad \cdot P[J_{m,n}=j_{m,n},I_{m,n}=l|I_{m,n-1}=k,\mathbf{j}^{(-\underline{j_m})}] \\
&= \sum_{k=1}^{N_m} P[j_{m,1},j_{m,2},\ldots,j_{m,n-1},I_{m,n-1}=k,\mathbf{j}^{(-\underline{j_m})}] \\
&\quad \cdot P[J_{m,n}=j_{m,n},I_{m,n}=l|I_{m,n-1}=k,\mathbf{j_{m,n}}] \\
&= \sum_{k=1}^{N_m} \alpha_{m,n-1}(k)P[J_{m,n}=j_{m,n}|I_{m,n}=l]P[I_{m,n}=l|I_{m,n-1}=k,\mathbf{j_{m,n}}] \\
&= \{\sum_{k=1}^{N_m} \alpha_{m,n-1}(k)P[I_{m,n}=l|I_{m,n-1}=k,\mathbf{j_{m,n}}]\}P[J_{m,n}=j_{m,n}|I_{m,n}=l], \\
&\quad l=1,\ldots,N_m, \qquad n=2,\ldots,T
\end{aligned} \tag{2.8}
$$

where $P[I_{m,1}=l|\mathbf{j_{m,1}}]$ and $P[I_{m,n}=l|I_{m,n-1}=k,\mathbf{j_{m,n}}]$ are obtained from equation 2.1 for our proposed method.

Let us consider the backward probability at this point. By using the same conditional independence assumption as in the forward case, we can write $P[I_{m,n+1}=k|I_{m,n}=l,\mathbf{j}]=P[I_{m,n+1}=k|I_{m,n}=l,\mathbf{j_{m,n+1}}]$. This assumption is also used to further simplify our backward recursive equation. The recursive equation for backward

probability is as follows,

$$
\begin{aligned}
\beta_{m,T}(l) &= 1, \qquad l = 1, \ldots, N_m \\
\beta_{m,n}(l) &= \sum_{k=1}^{N_m} P[j_{m,n+1}, j_{m,n+2}, \ldots, j_{m,T}, I_{m,n+1} = k | I_{m,n} = l, \mathbf{j}^{(-\underline{j_m})}] \\
&= \sum_{k=1}^{N_m} P[I_{m,n+1} = k | I_{m,n} = l, \mathbf{j}^{(-\underline{j_m})}] \\
&\quad \cdot P[j_{m,n+1}, j_{m,n+2}, \ldots, j_{m,T} | I_{m,n+1} = k, \mathbf{j}^{(-\underline{j_m})}] \\
&= \sum_{k=1}^{N_m} P[I_{m,n+1} = k | I_{m,n} = l, \mathbf{j_{m,n+1}}] \\
&\quad \cdot P[j_{m,n+1}, j_{m,n+2}, \ldots, j_{m,T} | I_{m,n+1} = k, \mathbf{j}^{(-\underline{j_m})}] \\
&= \sum_{k=1}^{N_m} P[I_{m,n+1} = k | I_{m,n} = l, \mathbf{j_{m,n+1}}] \\
&\quad \cdot P[J_{m,n+1} = j_{m,n+1} | I_{m,n+1} = k] P[j_{m,n+2}, \ldots, j_{m,T} | I_{m,n+1} = k, \mathbf{j}^{(-\underline{j_m})}] \\
&= \sum_{k=1}^{N_m} P[I_{m,n+1} = k | I_{m,n} = l, \mathbf{j_{m,n+1}}] P[J_{m,n+1} = j_{m,n+1} | I_{m,n+1} = k] \\
&\quad \cdot \beta_{m,n+1}(k), \qquad l = 1, \ldots, N_m, \qquad t = T-1, \ldots, 1
\end{aligned} \tag{2.9}
$$

where $P[I_{m,n+1} = k | I_{m,n} = l, \mathbf{j_{m,n+1}}]$ is obtained from equation 2.1.

Once the α and β are calculated, the $P[I_{m,n} = l|\mathbf{j}]$ can be calculated using (2.7). It is apparent that directly implementing these recursive equations would result in numerical underflow. Therefore, a renormalization step for both α and β must be done at each iteration to avoid this problem [49]. A conditional mean estimator[40] is then applied to estimate the quantization levels transmitted. This formulation above shows our proposed method has a slight computational complexity increase compared to a first

order SAMMSE decoder[40] due to the number of summations needed for estimating high order probability using IIS (2.1). It also shows our proposed method has a linear increase in memory complexity with model order increase. However, for a standard SAMMSE approach[40], computational and memory complexity increases exponentially with model order increase.

2.3 Experiment

In our experiments, we used 10 training images and 1 test image with size of $512X512$. There are two stages in our experiments, we first train our high order Markov model; then test our algorithm outside the training set.

Let us consider training for the Markov model at this point. Each image was first linear transformed by using a 2D DCT with $8X8$ blocks. This creates 64 transformed sources sorted from the highest to lowest variance with 40960 elements in each one. We set the overall rate of 0.125 bits per symbol, where only five sources ranked from highest variance are given the individual rate of $\{3, 2, 1, 1, 1\}$ with the rest of source rates set to zero. Each source is encoded (quantized) using a 4 state TCQ with its given bit rate. The source encoded symbols are then transmitted through a (simulated) noisy channel with a set of bit error rates shown in Figure 2.2. The received noisy symbol sequence and the transmitted source symbol sequence for each source are then used for training our high order hidden Markov model using IIS algorithm with second order constraints obtained by frequency counts as shown in section 2.2.2. Each source uses a different context space, with its conditional probability specified as follows (m=1 starts from top,

n=1 starts from left),

$$
\begin{aligned}
P[I_{m,n}|I_{m,n-1},\mathbf{j_{m,n}}] &= P[I_{m,n}|I_{m,n-1},j_{m+1,n-1},j_{m+1,n},j_{m+1,n+1},j_{m+2,n-1},\\
&\quad j_{m+2,n},j_{m+2,n+1},j_{m+3,n-1},j_{m+3,n},j_{m+3,n+1}] \quad m=1,\\
P[I_{m,n}|I_{m,n-1},\mathbf{j_{m,n}}] &= P[I_{m,n}|I_{m,n-1},j_{m-1,n-2},j_{m-1,n-1},j_{m-1,n},j_{m-1,n+1},\\
&\quad j_{m-1,n+2}] \quad m=2,\\
P[I_{m,n}|I_{m,n-1},\mathbf{j_{m,n}}] &= P[I_{m,n}|I_{m,n-1},j_{m-2,n-1},j_{m-2,n},j_{m-2,n+1},j_{m-1,n-1},\\
&\quad j_{m-1,n},j_{m-1,n+1}] \quad m=3,\\
P[I_{m,n}|I_{m,n-1},\mathbf{j_{m,n}}] &= P[I_{m,n}|I_{m,n-1},j_{m-3,n-1},j_{m-3,n},j_{m-3,n+1},j_{m-2,n-1},\\
&\quad j_{m-2,n},j_{m-2,n+1},j_{m-1,n-1},j_{m-1,n},j_{m-1,n+1}] \quad m=4,\\
P[I_{m,n}|I_{m,n-1},\mathbf{j_{m,n}}] &= P[I_{m,n}|I_{m,n-1},j_{m-4,n-1},j_{m-4,n},j_{m-4,n+1},j_{m-3,n-1},\\
&\quad j_{m-3,n},j_{m-3,n+1},j_{m-2,n-1},j_{m-2,n},j_{m-2,n+1},j_{m-1,n-1},\\
&\quad j_{m-1,n},j_{m-1,n+1}] \quad m=5. \qquad (2.10)
\end{aligned}
$$

From this setup, we used as high as 13th order in the 5th source with only one clean symbol $I_{m,n-1}$ in the conditioning contexts for all five sources. The noisy symbols in the conditioning context are what we defined earlier as the effective noisy symbol neighborhood $\mathbf{j_{m,n}}$.

At this point, let us consider testing the algorithm after the Markov model has been learned. We first source encode our test image by using the quantizers designed from training. The source encoded symbols are then transmitted through a noisy channel with a bit error rate the same as in training. This received noisy symbol sequence

is being used to estimate the transmitted quantization levels in our JSC decoder as shown in section 2.2.3. The reconstructed quantization levels in each source are then inverse DCT transformed to obtain the estimated gray scale values for the original image. Average peak signal to noise ratio ($10 \log_{10} \frac{255^2}{\text{Average MSE}}$) is measured after inverse DCT transformation. This average peak signal to noise ratio is calculated over five channel realizations.

2.4 Results

In Figure 2.2, several experiment results are included for comparison. "$P[I_{m,n}|I_{m,n-1}]$ freq count" is a first order SAMMSE decoder with its model parameters learned by frequency counts. "$P[I_{m,n}|I_{m,n-1}, \mathbf{j_{m,n}}]$ iter scale" is our proposed high order SAMMSE decoder with conditioning context including both clean and noisy symbols specified in equation 2.10. This high order model is achieved by using the IIS algorithm. "$P[I_{m,n}|I_{m,n-1}, \mathbf{j_{m,n}}]$ freq count reduced method" is the SAMMSE decoder with its model parameters learned by frequency counts but its order is reduced from equation 2.10 until no empty cell occurs in every joint PMF at each source. "$P[I_{m,n}|I_{m,n-1}, \mathbf{j_{m,n}}]$ freq count 4th order" is a 4th order SAMMSE decoder for each source with its model parameters learned by frequency counts. "$P[I_{m,n}|I_{m,n-1}, \mathbf{j_{m,n}}]$ freq count 3rd order" is a 3rd order SAMMSE decoder for each source with its model parameters learned by frequency counts. "$P[I_{m,n}|I_{m,n-1}, \mathbf{j_{m,n}}]$ freq count 2nd order" is a 2nd order SAMMSE decoder for each source with its model parameters learned by frequency counts. Note that $P[I_{m,n}|I_{m,n-1}, \mathbf{j_{m,n}}]$ indicates the model parameters including one clean symbol

$I_{m,n}$ and effective noisy symbol neighborhood $\mathbf{j_{m,n}}$ in its conditioning context. Also, the order of decoder is the number of elements in the conditioning context.

Let N_f denote the number of elements in the conditioning context of a model. As shown in Table 2.1 and Figure 2.2, our proposed algorithm has only slightly higher computational complexity compared to the first order SAMMSE, but we were able to gain as much as about 1dB for the bit error rates we tested. In fact, for $N = 32$ states, $T = 4096$ symbols, and $N_f = 13$ symbols we tested, the computational complexity of our high order decoder is only approximately 1.5 *times* in number of sums and *the same* in number of multiplications compared to a first order SAMMSE decoder, while the computational complexity of the same order SAMMSE decoder (using only hidden states/clean symbols in its conditioning context) would incur approximately 32^{12} *times* increase in both number of sums and multiplications. Although our memory storage ($N^2 N_f$) increases linearly with model order increase, the memory storage of a standard SAMMSE decoder (N^{N_f+1}) increases exponentially. In Figure 2.2, it shows that the decoding performance our proposed method is superior than all other methods with similar computational complexity. This is largely due to the fact that our method learns its maximum entropy high order model from a large group of second order constraints, while the other methods learn their model through direct (high order) frequency counts, which are less accurate, and hence which introduce overfitting. Given 10 training images, other methods suffer from the overfitting problem. This problem can be seen in Table 2.1, where the memory storage requirement increases exponentially with model order increase for other methods. Another thing to note in Figure 2.2 is that the "$P[I_{m,n}|I_{m,n-1}, \mathbf{j_{m,n}}]$ freq count 3rd order" decoder performs better than all other ones we tried except our

proposed method. This is because given enough training data, a higher order SAMMSE decoder is suppose to perform better than the lower order ones since more statistical redundancy can be captured. However, since we used 10 training images, this result indicates the overfitting problem becomes severe when the order of decoder is higher than 3. Hence, "$P[I_{m,n}|I_{m,n-1}, \mathbf{j_{m,n}}]$ freq count 3rd order" decoder performs better than "$P[I_{m,n}|I_{m,n-1}, \mathbf{j_{m,n}}]$ freq count 4th order" decoder.

	1^{st} order SAMMSE *(clean)*	Our Method	2^{nd} order SAMMSE *(clean)*	freq. count reduced method	freq. count 2^{nd} order	freq. count 3^{rd} order	freq. count 4^{th} order
Multi.	N^2T	N^2T	N^3T	N^2T	N^2T	N^2T	N^2T
Sum	N^2T	$N(N+N_f)T$	N^3T	N^2T	N^2T	N^2T	N^2T
memory storage	N^2	N^2N_f	N^3	N^{N_f+1}	N^3	N^4	N^5

Table 2.1. complexity comparison between methods

2.5 Conclusion

In this chapter, we have proposed a novel JSC decoding technique which achieves performance-complexity tradeoffs that fall between those of low and high order JSC decoders. There are two major contributions associated with our technique. One, the computational complexity of our higher order decoder only increases slightly compared to a standard first order SAMMSE decoder. By contrast, the computational complexity increases exponentially with model order increase in a standard SAMMSE decoder. The

computational complexity of a standard SAMMSE decoder with the order we tried (13) would exceed by far the capability of modern computers. Two, we reduced the substantially large training set requirement needed for high order Markov model by using IIS algorithm learning a constrained high order model. In our experiments, we used 10 training images. *10 thousand* training image would be needed to keep the number of elements per each joint PMF frequency cell roughly the same if directly using frequency counts. Obviously, this is not practical. Additionally, the memory complexity of our proposed method only increases *linearly* with model order increase, while the memory complexity of a standard SAMMSE decoder increases *exponentially*.

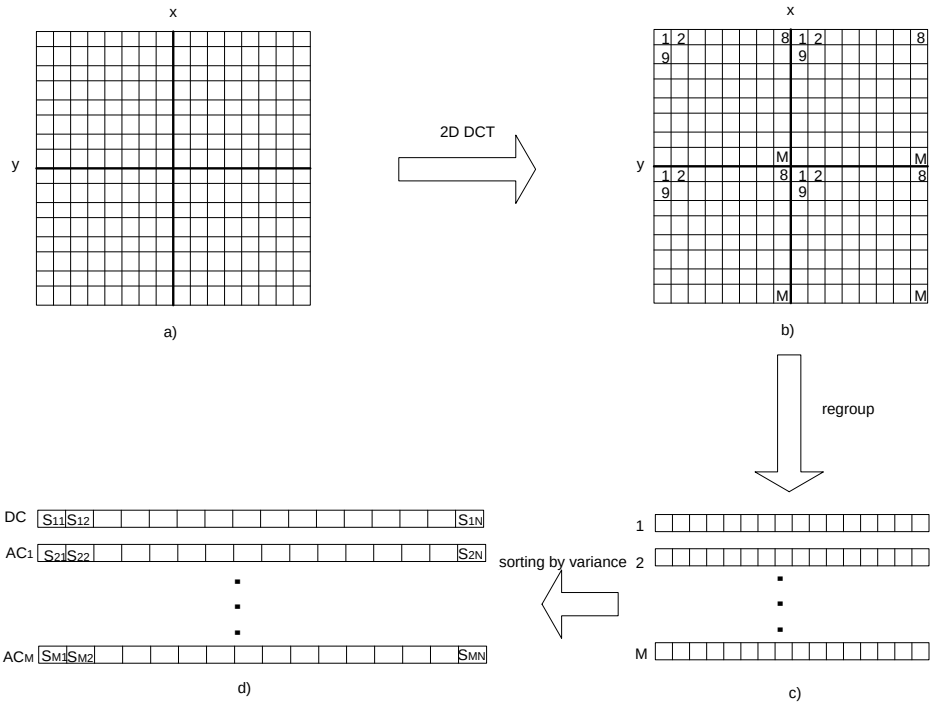

Fig. 2.1. Image transformation and sources rearranged procedure. a)Divide image into blocks b) After 2D DCT, coefficients are combined. c)Group coefficient at the same frequency to form a sequence d)Sort these sequences from highest variance(DC) to lowest(AC_M)

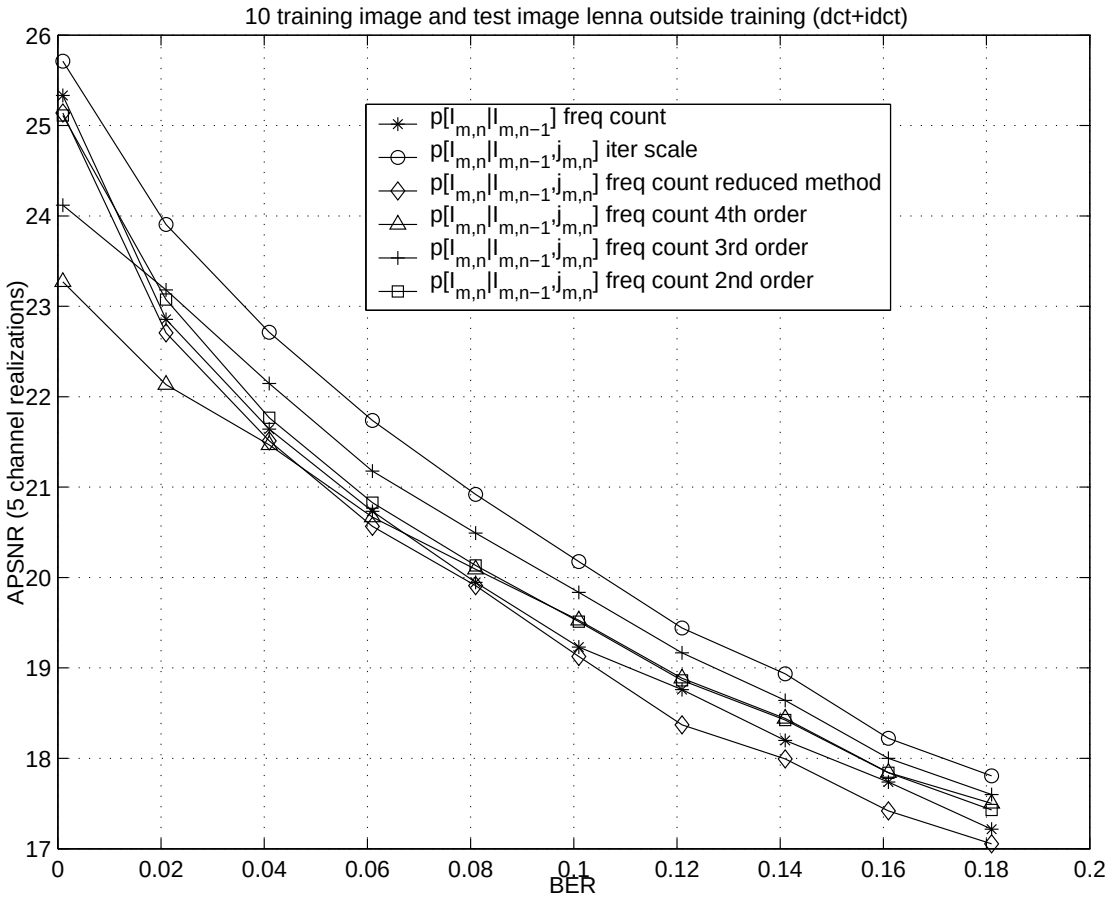

Fig. 2.2. Decoding performance (average PSNR in dB over 5 channel realizations)

Chapter 3

A Two-Stage Estimation Approach for Bridging the Performance-Complexity Gap in Joint Source-Channel Decoding

3.1 Introduction

Both maximum *a posteriori* (MAP) [48] and minimum mean-squared error (MMSE) [40] techniques have been proposed for JSC decoding. These JSC decoding techniques have been applied both to non-predictive coding systems (e.g. scalar and vector quantization) [40, 43, 48] as well as to predictive coding systems (e.g. DPCM) [14, 31, 46, 54]. They have also been applied both to time series and to images. One difficulty with these methods, especially when applied to images, is the computational and storage complexity of the decoding. As mentioned in the previous chapter, in either case, the decoding complexity increases *exponentially* with the Markov model order K, i.e. the complexity is $O(N^{K+1}T)$, with N the cardinality of the state and T the sequence length. Thus, in the literature, one typically only sees low orders investigated, e.g. K=2 or 3. In the case of an image source, the computational difficulties are only accentuated. In [46] a Markov mesh model was suggested for a DPCM encoded source. Even for the most minimal causal conditioning context, based on one pixel to the left and one pixel above the current pixel of interest, the complexity of exact a posteriori state estimation goes as $O(N^5T)$. In [47], approximate state estimation was performed, with $O(N^3T)$ complexity. Clearly, as the conditioning context increases, exact state estimation becomes

practically intractable. Even with high order conditioning contexts, there are ways of keeping the decoding complexity manageable, but at the price of estimation accuracy. One practical solution is to use greedy techniques, e.g. [14], where hard state estimation was performed "one pixel at a step". In this approach, when estimating the state at a current pixel of interest, one uses as conditioning context the decoded values in a causal neighborhood of this pixel. The *storage* complexity of this approach will *still* grow exponentially with conditioning order. However, the computational complexity is now only *linear* in the state cardinality, i.e. $O(NT)$. A related *soft* state estimation technique was proposed in [31]. Here, *a posteriori* state probabilities are computed "one pixel at a step", using previously computed *a posteriori* probabilities in the causal neighborhood of the current pixel. However, unlike the "hard" case[14], the decoding complexity for the approach in [31] *still* grows exponentially, in particular $O(N^{K+1}T)$, with K the (causal) conditioning order. There are also some techniques for improving upon hard greedy searches. In [7], an iterative hillclimbing algorithm was proposed with guaranteed improvement over greedy search. In this approach, instead of optimizing "one pixel at a step", one jointly optimizes over an entire row or column at each step, achieved via dynamic programming. Moreover, whereas the greedy search in [14] terminates after one pass over the image, in [7] the solution can be iterated, with multiple row/column sweeps taken over the image, and with the performance guaranteed to improve (in the sense of the defined objective function) with each image pass. The complexity of this method is $O(NTL)$, with L the number of image iterations taken. From the discussion above, we can see that there is a fundamental difference in complexity in the greedy hard and soft estimation cases. Computation grows linearly in N in the hard case and

as N^K in the soft case. However, it is also well-known that soft estimation techniques generally outperform their hard counterparts. The reason being that they preserve many hypotheses for the hidden state sequence (and attach associated probabilities), whereas hard state estimation preserves only one such hypothesis. In this chapter, we derive a general decoding technique that aims to capture some of the advantage of higher order soft estimation while still retaining low complexity. We thus suggest an approach which can *bridge* the performance gap between low order (low complexity) and high order (high complexity) JSC decoding. Our decoding approach consists of two stages: 1) low order JSC decoding, followed by 2) a linear FIR filtering of the JSC decoded signal. The linear filter is chosen to provide an optimal (least squares) estimate of the original source. The new approach is demonstrated to significantly improve upon standard MMSE-based JSC decoding performance, both for the case of nonpredictive source coding (e.g. vector quantization) as well as for predictive source coding (DPCM).

3.2 New JSC Decoding Formulation

Preliminaries

Consider the problem of source coding, with transmission of the encoded information over a noisy channel. The source encoder takes $X_t \in \mathcal{R}$ and produces an output index $I_t \in \mathcal{I}$, with $\mathcal{I}$ the quantization index set $\{0, 1, \ldots, N-1\}$. The index is transmitted over a discrete channel, resulting in a possibly corrupted index J_t, from the same set $\mathcal{I}$, according to the (assumed constant) channel transition probabilities $\{P[J_t = j_t | I_t = i_t]\}$. We define the data source, from a discrete-time random process, as the sequence

$\underline{X} \equiv (X_0, X_1, \ldots, X_{T-1})$, the encoder's index sequence $\underline{I} \equiv (I_0, I_1, \ldots, I_{T-1})$, and the received sequence, $\underline{J} \equiv (J_0, J_1, \ldots, J_{T-1})$. As in prior work [40, 48, 54], we assume a Markovian model for the sequence of transmitted indices, e.g. in the first order case,

$$P[I_0 = i_0, I_1 = i_1, \ldots, I_t = i_t] = P[I_0 = i_0] \prod_{l=1}^{t} P[I_l = i_l | I_{l-1} = i_{l-1}], \quad \forall t. \tag{3.1}$$

All the probabilities $\{P[J_t = j_t | I_t = i_t]\}$, $\{P[I_0 = i_0]\}$, $\{P[I_t = i_t | I_{t-1} = i_{t-1}]\}$ are assumed known at the decoder. In practice, the source probabilities are estimated based on an encoded training set. In the case of a binary symmetric channel, the channel is completely specified by a single parameter, ϵ, the channel bit error rate. The JSC decoder objective is to produce an approximation of the source $\underline{X}$, denoted $\underline{\hat{x}}^{(\mathrm{dec})} \equiv (\hat{x}_0^{(\mathrm{dec})}, \hat{x}_1^{(\mathrm{dec})}, \ldots, \hat{x}_{T-1}^{(\mathrm{dec})})$, given a realization of the noisy index sequence, $\underline{j} \equiv (j_0, j_1, \ldots, j_{T-1})$ [1]. The knowledge brought to bear by the decoder consists of the source and channel probability models.

The approach that we suggest builds on/*leverages* previous decoding techniques, such as [40]. The reader is referred to chapter 1 for detailed formulation of the SAMMSE decoder[40]. There are several ways to improve SAMMSE decoding. One strategy is to increase the order of the Markov model for $\{I_t\}$. However, the complexity of the forward and backward recursions needed to calculate the *a posteriori* probabilities grows exponentially with the Markov order K, i.e. the complexity is $O(N^{(K+1)}T)$. This limits

[1] Without knowledge of $\underline{j}$, the decoder output $\hat{X}_t^{(\mathrm{dec})}$ is treated as a random variable. However, the decoding rule is a deterministic function of the received sequence $\underline{j}$. For purpose of retaining compact notation, we do not explicitly indicate the dependence on $\underline{j}$ in $\underline{\hat{x}}^{(\mathrm{dec})}$.

the practical feasibility of this approach. A second potential strategy is to use "memory-enhanced" decoding, as suggested in [40]. This approach uses a higher resolution decoder lookup table and thus obtains a more accurate conditional mean estimate. For the case of second order (enhanced memory) decoding, the decoding rule is:

$$E[X_t|\underline{j}] = \sum_l \sum_m Q_{\text{dec}}^{-1}(l,m) P[I_t = l, I_{t-1} = m|\underline{j}], \tag{3.2}$$

where $Q_{\text{dec}}^{-1}(l,m) \equiv E[X_t|I_t = l, I_{t-1} = m]$. It should be emphasized that the decoder memory can be usefully chosen to be second or higher order *irrespective of* the order of the Markov model for $\{I_t\}$ [2]. Unfortunately, the number of summations increases linearly, and thus the number of summands *exponentially*, with the order of the decoder memory. Moreover, the complexity of calculating the *a posteriori* probabilities $P[I_t, I_{t-1}, \ldots, I_{t-K_d+1}|\underline{j}]$, *also* increases exponentially with the decoder memory.

While both increasing the Markov order and increasing the decoder memory improve performance, there is a heavy price in complexity. In the next section, alternatively, we will demonstrate a way of improving the SAMMSE decoder with only *modest* increases in complexity.

A New JSC Decoder

As just discussed, JSC decoding can be improved by increasing the Markov order or the decoder's memory, albeit with exponential growth in complexity. Alternatively, here we suggest a heuristic way to increase the decoder "order" with minimal growth in

[2] It is also not necessary for the conditioning context to be causal. It could also be noncausal, e.g. $E[X_t|I_t = m, I_{t-1} = l, I_{t+1} = n]$.

complexity. In particular, we suppose that if the decoder model is not fully capturing the memory in the source, then it may very well be the case that there is unexploited *partial correlation* between the source X_t and both causal $\{\hat{X}_{t-k}^{(\mathrm{dec})}, k > 0\}$ and *anticausal* $\{\hat{X}_{t+k}^{(\mathrm{dec})}, k > 0\}$ decoded values. In making this statement, we are recognizing that $\hat{X}_{t-k}^{(\mathrm{dec})}(\underline{J})$ is truly a *random* quantity, as it is a function of the random sequence $\underline{J}$. If there is untapped correlation, this suggests the possibility of improving the decoding result $\hat{X}_t^{(\mathrm{dec})}$ via linear filtering, i.e. forming

$$\hat{x}_t^{(\mathrm{newdec})} = \sum_{k=0}^{L_{\mathrm{c}}-1} \alpha_k \hat{x}_{t-k}^{(\mathrm{dec})} + \sum_{k=1}^{L_{\mathrm{nc}}} \alpha_{-k} \hat{x}_{t+k}^{(\mathrm{dec})}. \tag{3.3}$$

(3.3) amounts to a *two-stage* estimation procedure, with standard SAMMSE decoding first applied, followed by linear filtering. Low order (e.g. $K = 1$) SAMMSE decoding followed by the filtering in (3.3) is *far* less complex than SAMMSE decoding based on a high order Markov model. The complexity comparison will be further discussed in the experimental result section. It remains to determine the coefficients $\{\{\alpha_k\}, \{\alpha_{-k}\}\}$.

In JSC decoding, e.g. [40, 48, 54], the approach often taken is to assume an optimality criterion (such as MMSE or MAP) and a statistical model and then to analytically derive a closed form decoder expression. Alternatively, we take our cue from *training-based* approaches to quantizer design, proposing a training-based approach to the design of a JSC decoder that captures residual redundancy in the source. The ultimate performance is the MSE $E[(X - \hat{X}^{(\mathrm{dec})})^2]$, with the expectation taken with respect to both the source and channel distributions. Now, as is often done in practice without

proof, e.g. [27], let us suppose that an *ergodicity property* holds, i.e., in our case, that

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} (x_t - \hat{x}_t^{(\mathrm{dec})}(\underline{j}))^2 = E[(X - \hat{X}^{(\mathrm{dec})})^2], \tag{3.4}$$

where we have emphasized the rule's dependence on $\underline{j} = (j_0, j_1, \ldots, j_{T-1})$. The motivation behind this is that, if (3.4) holds, then one can choose the rule $\hat{x}_t^{(\mathrm{dec})}(\underline{j})$ to minimize a least squares cost based on a large training set *and a single realization of the channel*, $\underline{j}$, with the reasonable expectation that one is then (approximately) choosing the decoder to minimize the MSE $E[(X - \hat{X}^{(\mathrm{dec})})^2]$. A similar approach was taken in [9] for a different source coding context and for optimization at the *encoder*, not the decoder. The reasonableness of our ergodicity assumption will be substantiated by our experimental results.

Thus, the coefficients $\{\{\alpha_k\}, \{\alpha_{-k}\}\}$ can be chosen to minimize the *least squares error* (LSE) performance criterion,

$$\sum_{t=L_c-1}^{T-L_{nc}-1} (x_t - \hat{x}_t^{(\mathrm{dec})}(\underline{j}))^2. \tag{3.5}$$

The LS-optimal coefficients are given by the standard LS solution:

$$\underline{\alpha} = (\mathcal{X}^T \mathcal{X})^{-1} \mathcal{X}^T \underline{x}, \tag{3.6}$$

with

$$\mathcal{X}^T = \begin{bmatrix} \hat{x}_0^{(\mathrm{dec})} & \hat{x}_1^{(\mathrm{dec})} & \cdots & \hat{x}_{T-L_c-L_{nc}}^{(\mathrm{dec})} \\ \hat{x}_1^{(\mathrm{dec})} & \hat{x}_2^{(\mathrm{dec})} & \cdots & \hat{x}_{T-L_c-L_{nc}+1}^{(\mathrm{dec})} \\ \vdots & \vdots & \vdots & \vdots \\ \hat{x}_{L_c+L_{nc}-1}^{(\mathrm{dec})} & \hat{x}_{L_c+L_{nc}}^{(\mathrm{dec})} & \cdots & \hat{x}_{T-1}^{(\mathrm{dec})} \end{bmatrix}, \tag{3.7}$$

$\underline{\alpha} = (\alpha_{L_c-1}, \alpha_{L_c-2}, \alpha_{L_c-3}, \cdots, \alpha_0, \alpha_{-1}, \cdots, \alpha_{-L_{nc}})^T$,

and $\underline{x} = (x_{L_c-1}, x_{L_c}, \cdots, x_{T-L_{nc}-1})^T$ obtained from a training sequence. The resulting solution can be written in the (explicit) form:

$$\begin{aligned} \hat{x}_t^{(\mathrm{newdec})} = & \sum_{k=0}^{L_c-1} \alpha_k \Big(\sum_{l=0}^{N} P[I_{t-k} = l|\underline{j}] y(l)\Big) \\ & + \sum_{k=1}^{L_{nc}} \alpha_{-k} \Big(\sum_{l=0}^{N} P[I_{t+k} = l|\underline{j}] y(l)\Big), \end{aligned} \tag{3.8}$$

where $y(l) \equiv E[X|I = l]$.

This form is suggestive of a way to further *improve* the solution. In particular, the quantities $y(l) \equiv E[X|I = l]$ are usually estimated based on an encoded training set. Alternatively, we can replace the products $\alpha_k y(l)$ by the *parameters* $\mu_{k,l}$, with the decoding rule now

$$\hat{x}_t^{(\mathrm{newdec})} = \sum_{k=0}^{L_c-1} \sum_{l=0}^{N-1} \mu_{k,l} P[I_{t-k} = l|\underline{j}] + \sum_{k=1}^{L_{nc}} \sum_{l=0}^{N-1} \mu_{-k,l} P[I_{t+k} = l|\underline{j}], \tag{3.9}$$

and with the $\{\mu_{k,l}\}$ again chosen to minimize the least squares cost (3.5). The performance of both (3.8) and (3.9) are evaluated in our experimental results. Next, we briefly discuss the implications of our LS approach for *predictive* JSC decoding.

JSC Decoding for Predictively Encoded Sources

In [14, 31, 46, 54], a common JSC decoding strategy for predictively encoded sources (e.g. DPCM) is to first form a JSC decoding estimate of the prediction residual and then feed this estimate to a standard (noise-free) DPCM decoder. For example, in [46], for the case of first order DPCM, the proposed decoding rule was:

$$\hat{x}_t^{(\mathrm{dec})} = a\hat{x}_{t-1}^{(\mathrm{dec})} + E[Z_t|\underline{j}], \tag{3.10}$$

where $E[Z_t|\underline{j}] = \sum_{l=0}^{N-1} c(l)P[I_t = l|\underline{j}]$ and $c(l) \equiv E[Z_t|I_t = l]$. Here, Z_t is the prediction residual and a is the prediction coefficient. Note that $E[Z_t|\underline{j}]$ is just the SAMMSE decoding estimate of the prediction residual.

The main principle behind our LS decoding strategy is to treat the standard JSC decoding outputs $\hat{X}_t^{(\mathrm{dec})}(\underline{J})$ as *data*, i.e., as random observations, correlated with the true signal X_t. Accordingly, in the last section we designed linear filters to provide an LS estimate of X_t given $\{\hat{X}_t^{(\mathrm{dec})}(\underline{J})\}$. This approach improves performance at the cost of *some* increase in implementation complexity, associated with the linear filtering. However, the complexity increase is modest compared with increasing the Markov order or using memory-enhanced decoding[40]. In the predictive coding case, use of our LS strategy can yield improved decoders with *no* complexity increase, i.e. a fundamental improvement can be achieved using our LS decoding strategy. Alternatively, further gains

can again be achieved with modest increase in complexity. Our LS design strategy for JSC decoding of DPCM-encoded sources was proposed in [39]. Here, we summarize those results. The key observation is that the predictive JSC decoding rule (3.10) is *already* in the form of a linear estimator, based on the "observations" $\hat{x}_{t-1}^{(\mathrm{dec})}$ and $E[Z_t|\underline{j}]$. This raises the question of whether the coefficients $(a, 1)$ are *optimal* (or nearly so) in the least squares sense, as the weights for these observations.

Consider the more general estimator

$$\hat{x}_t^{(\mathrm{newdec})} = \alpha \hat{x}_{t-1}^{(\mathrm{dec})} + \beta E[Z_t|\underline{j}], \tag{3.11}$$

with $\hat{x}_{t-1}^{(\mathrm{dec})}$ obtained from the standard decoding rule (3.10). I.e., suppose (3.10) is first applied, yielding $\{\hat{x}_t^{(\mathrm{dec})}\}$, with the optimal coefficients then sought to weight the values $\hat{x}_{t-1}^{(\mathrm{dec})}$ and $E[Z_t|\underline{j}]$ in forming a *new* decoding estimate $\hat{x}_t^{(\mathrm{newdec})}$. For a training set $\underline{x} = (x_0, x_1, \ldots, x_{T-1})^T$ and a realization of the channel $\underline{j}$, the pair (α, β) optimal in the least squares sense again satisfies

$$(\alpha \ \beta)^T = (\mathcal{X}^T \mathcal{X})^{-1} \mathcal{X}^T \underline{x}, \tag{3.12}$$

with

$$\mathcal{X}^T = \begin{bmatrix} \hat{x}_{-1}^{(\mathrm{dec})} & \hat{x}_0^{(\mathrm{dec})} & \cdots & \hat{x}_{T-2}^{(\mathrm{dec})} \\ E[Z_0|\underline{j}] & E[Z_1|\underline{j}] & \cdots & E[Z_{T-1})|\underline{j}] \end{bmatrix}, \tag{3.13}$$

and with $\hat{x}_{-1}^{(\mathrm{dec})}$ an initial value. Our approach to JSC decoding for predictively encoded sources, similar to the approach for nonpredictive coding, is to use the LS-optimal coefficients in the decoding rule (3.11).

Consider an example of a first order Gauss-Markov process $\{X_t\}$ with correlation coefficient 0.95, $N = 8$ quantization levels, a first order Markov model for the sequence $\{I_t\}$, and a binary symmetric channel with bit error rate $\epsilon = 0.05$. Residual redundancy is introduced by mismatching the prediction coefficient, relative to the source correlation. Suppose the prediction coefficient is chosen as $a = 0.45$. Experimentally, we have found that this choice leads to good decoding performance both for our system and for the standard JSC decoder (which uses $(a, 1)$). For this choice, we find that the LS-optimal pair (based on a training set of 10^6 samples), is $(\alpha, \beta) = (0.55, 0.79)$, quite different from the values $(0.45, 1)$ that give the standard rule. Morever, averaged over 3 test sets of size fifty thousand samples, the reduction in distortion of this LS-optimal decoder over the standard JSC decoder is $10\log 10$(MSE of standard JSC/MSE of new JSC) = 0.41 dB. Clearly, the standard JSC rule is *not* in general the optimal way to combine the "observations" $E[Z_t|\underline{j}]$ and $\hat{x}_{t-1}^{(\mathrm{dec})}$.

Even better performance is achieved by increasing the order of the filter, i.e. forming:

$$\hat{x}_t^{(\mathrm{newdec})} = \sum_{k=1}^{L_\mathrm{c}} \alpha_k \hat{x}_{t-k}^{(\mathrm{dec})} + \beta E[Z_t|\underline{j}],\ L_\mathrm{c} \geq 1. \tag{3.14}$$

Moreover, while (3.14) only includes causal terms, our method can also be applied to design a decoder that uses *anticausal* terms $\{\hat{x}_{t+k}^{(\mathrm{dec})}, k \geq 0\}$, with the potential for additional gains in performance. In the next section, we evaluate the LS design approach we

have developed here (and the various nonpredictive and predictive decoding structures) in comparison with conventional JSC decoding approaches.

3.3 Experimental Results

We investigated our LS optimization approach for improving the performance of first order SAMMSE decoding. As a source, we chose the first AC (AC1) DCT coefficient from 8x8 blocks of a gray scale image. This one dimensional AC1 source was created by 1) DCT transforming the image using 8x8 blocks, 2)collecting the same transform coefficients from each of the 8x8 blocks to form 64 different coefficient "images", 3)for each image, a 1D source is created by raster scanning; We chose to encode the source with second highest variance (AC1). As a source encoder, we used a scalar quantizer with eight quantization levels (3 bit fixed length). This quantizer was designed via the Lloyd algorithm using the AC1 source extracted from 23 gray scale images. We chose the bit error rate of our binary symmetric channel to be 0.05. Both the first and second order SAMMSE decoders were implemented based on these images. The Markov model probabilities for these decoders were learned based on frequency counts from the AC1 coefficients for all 23 images. The first order SAMMSE decoder gave an average SQNR($10log_{10}\frac{\sigma_x^2}{\text{Average MSE}}$) of 2.91 dB over this 23 image source. The second order SAMMSE decoder based on the same source, gave an average SQNR of 4.52 dB.

For our method, we first designed an LS-optimal filter for the first order SAMMSE decoding result based on the decoder form (3.8). Our decoding performance was tested

on all 23 images with LS filter coefficients optimized for each image [3]. The result is shown in Figure 3.1. In this figure, the 'number of filter' taps include both causal and anticausal coefficients. The number of filter taps increases by adding one tap from causal and one tap from anticausal context alternately, initialized by one causal filter tap. Let us consider the case when $L_c = 3$ and $L_{nc} = 2$. With these filter taps, only a small amount of side information is needed to specify the filter coefficients to the decoder for each image. The resulting averaged LS-optimal performance was 4.12 dB. This is approximately 1.2 dB better than using only a first order SAMMSE decoder, and is only approximately 0.4 dB worse than a second order SAMMSE decoder. In this case, the LS decoder complexity is roughly 8% higher than first order SAMMSE and roughly 7.4 times less complex than second order SAMMSE.

We also designed an LS-optimal filter that filters the *a posterior* probabilities of a first order SAMMSE decoding result based on the decoder form (3.9). The result is shown in Figure 3.2. In this figure, the number of filter taps also include both causal and anticausal coefficients. Each "tap" in this case corresponds to eight coefficients. These 8 coefficients are associated with eight possible decoded symbol values at each symbol time. The number of filter tap sets $\underline{\mu_k}$ increases in the same fashion as in Figure 3.1. Let us note the case when there is only one set of filter taps. The resulting averaged LS-optimal performance was 4.41 dB. This is approximately 1.5 dB better than using only a first order SAMMSE decoder, and is only approximately 0.1 dB worse than a second order SAMMSE decoder. In this case, the LS decoder complexity is roughly

[3] We found that only small performance gain was achieved if a single "universal" filter $\{\alpha\}$ is used for all the images

12.5% higher than first order SAMMSE and roughly 7.1 times less complex than second order SAMMSE. Compared to the same computational complexity with 8 filter taps in Figure 3.1, this approach is approximately 0.3 dB better.

In Table 3.1, a comparison between the standard predictive decoder(3.10) and the LS optimized predictive decoder(3.14) using 10 causal coefficients is shown. This predictive result is obtained from our joint work[39]. In this experiment, we used a first order Gauss-Markov source $\{X_t\}$ with correlation coefficient $\rho = 0.95$ and with input white noise variance $\sigma_w^2 = 1.0$. A training set of 1 million samples was used for this decoder design. Three independent test sets were generated, each of size 50,000 samples. We assumed first order DPCM at 3 bits/sample. A uniform quantizer was chosen, based on the dynamic range of the prediction residual. For SAMMSE estimation of the prediction residual $E[Z_t|\underline{j}]$, we assumed a first order Markov model for $\{I_t\}$. This table shows the performance gain of using LS optimization approach in the predictive case with different choices of prediction coefficient.

Prediction coefficient	Std. Dec. Avg. SQNR	LS Dec. Avg. SQNR	Gain vs Std. Dec.
0.35	11.761	12.533	0.772
0.45	12.008	12.539	0.531
0.55	11.987	12.354	0.367

Table 3.1. SQNR performance of standard and LS decoding for DPCM encoding of a Gauss-Markov source, as a function of the prediction coefficient. In this case, the LS decoder uses 10 causal samples.

3.4 Conclusions

Although higher order modeling could improve decoding performance, the direct implementation of high order (>3) SAMMSE decoding is typically too complex for modern computers. In this work, we proposed a method to bridge the performance-complexity gap between direct implementation of low and high order SAMMSE decoding. We showed that our LS filtering approach was able to increase decoding performance with a small increase in computational complexity. Our LS filter is chosen to provide an estimate of the original source via a large training set. This approach provides a way to approximate the effective order increase in the standard approach without the exponential increase in computational complexity. We have verified our approach by first designing a LS filter for the decoding form (3.8). We then further improved this approach by designing the LS filter for the form (3.9).

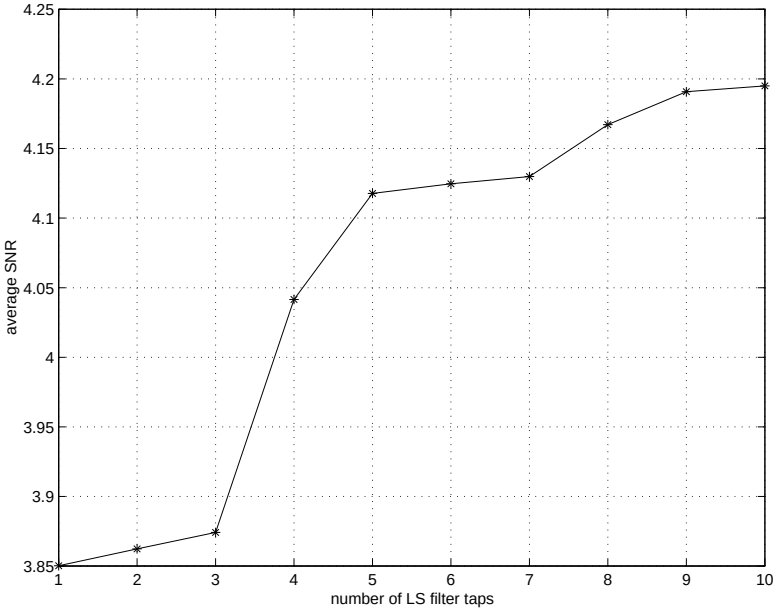

Fig. 3.1. LS filtering on $\hat{x}_t^{(\mathrm{dec})}$

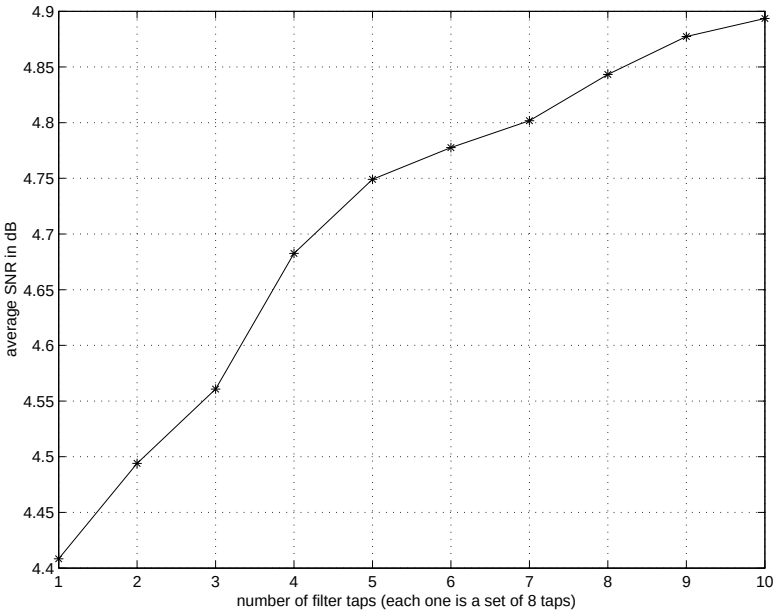

Fig. 3.2. LS filtering on *a posteriori* prob. $p[I_t = l|\underline{j}]$

Chapter 4

High Order Conditional Entropy-Constrained Encoding of Images via Improved Iterative Scaling

4.1 Introduction

Entropy constrained encoding chooses quantization levels to minimize distortion with the constraint of bit rate by using variable length coding[16, 35]. In an image encoding context, an entropy constrained encoding is done within each row (transformed source). However, the correlation between rows (transformed sources) of an image can be captured to further improve the encoding performance. One approach which tries to capture both correlation *within* rows/sources and *between* rows/sources is called conditional entropy-constrained encoding. This technique has been effectively applied in several coding contexts[10, 28]. A conditional entropy-constrained TCQ encoding (CECTCQ)[30] is widely used in image encoding due to its simplicity and channel error resilience. The CECTCQ algorithm uses TCQ[36] to minimize the overall mse distortion with rate constraint over quantization choices via generalized Lloyd algorithm (GLA)[33]. Its detailed formulation will be shown later in this chapter. Several methods have been proposed to improve the greedy CECTCQ algorithm[30]. In [11], the author proposed a soft-input-soft-output iterative optimization algorithm, alternatively applied to the image rows and columns. In [37], a rate-distortion optimized "bit flipping" algorithm was developed using multiple iterations over the image. This iteration process is done one pixel at a time.

In [7], an iterative hillclimbing algorithm was proposed to jointly optimize index assignments for entire rows/columns at each step with guaranteed non-decreasing Lagrangian cost at each iteration, which is done one row/column at a time.

In all these methods, increasing the order of the conditioning in the model translates to better encoding performance. This is due to the fact that more statistical dependence can be captured and used to further reduce the overall encoding bit rate. However, increasing the order of conditioning also means a large increase in training data size needed to ensure good learning of the model. This problem was also mentioned in chapter 2 (high order JSC image decoding with reduced complexity via improved iterative scaling) of this thesis. It was first addressed by Rissanen[52], and the term *model cost* was coined. Model cost is used to evaluate a statistical model for a source in a coding context. It includes the cost of describing/specifying the model to the decoder. Model cost can used to determine the overfitting problem when increasing order of conditioning model does not result in better modeling performance due to insufficient training data size for learning the model. Wu, et. al.[60] proposed a Bayesian-type technique which uses lower order constraints to solve this problem within the context of conditional entropy encoding of image vector quantization indices. In this chapter, we extend our idea of using the improved iterative scaling algorithm[4] from chapter 2 applying to the conditional entropy constrained encoding for images. We will show that we can achieve high order CECTCQ encoding for transformed images with only *linear* increase in training data set size needed as the model order increases. Our approach can also be applied to *any* high order conditioning context in the conditional entropy

constrained encoding framework to further improve encoding performance without needing a substantially large training data set. In the experimental results section, we will show a comparison between high order CECTCQ using our approach and a standard second order CECTCQ[30], based on the used of a fixed training set of 23 images. The justification that the comparison between these two methods is fair will be given in the result section. Our method has a PSNR ($log_{10}\frac{255^2}{\text{MSE}}$) gain of as much as 1 dB for some bit rates. Our proposed algorithm mitigates the curse of dimensionality problem. It also gives a practical approach to bridging the gap of performance-complexity tradeoff between direct learning of low and high order probability models via frequency counts.

4.2 Formulation

In this section, the detailed formulation is shown. For consistency, the setup of our DCT linear transformed images is the same as in chapter 2. The reader is also referred to chapter 1 for detailed development of TCQ and standard greedy conditional entropy constrained encoding. The development of the IIS algorithm, which is used to achieve high order modeling in the context of CECTCQ, is shown in section 4.2.1. The high order CECTCQ algorithm, which uses the high order model learned via IIS, is shown in section 4.2.2.

4.2.1 Improved Iterative Scaling Algorithm

In this chapter, we used second order probabilities as our lower order constraints which are measured from frequency counts using training data sets. Let $i_{m,n}$ denote the quantization index of current transformed value at m_{th} source and n_{th} element,

where $\underline{i_{m-1}}$ is the row/source above $\underline{i_m}$ and $\underline{i_{n-1}}$ is the column to the left of $\underline{i_n}$. The sources are sorted from highest variance ($m = 1$) to lowest. The high order neighborhood of $i_{m,n}$ is chosen empirically for each source/row and defined as $\mathbf{w_{m,n}}$. Due to the trellis structure in TCQ, each element $w_{q,r}$ in the neighborhood $\mathbf{w_{m,n}}$ is a two dimension vector $(i_{q,r}, \tilde{S}_{q,r})$, where $\tilde{S}_{q,r}$ denotes the TCQ state associated with $i_{q,r}$. The detailed configuration for this high order neighborhood will be shown in the experimental results section. By using IIS, we are learning the maximum entropy *a posteriori* probability $P_{ME}[I_{m,n} = l|\mathbf{w_{m,n}}]$ from a large group of pairwise constraints $\{P[I_{m,n} = l, W_{q,r} = w_{q,r}] \quad W_{q,r} \in \mathbf{w_{m,n}}\}$. All lower order probability constraints are enforced using Lagrange multipliers $\{\gamma(I_{m,n} = l, W_{q,r} = w_{q,r}) \quad W_{q,r} \in \mathbf{w_{m,n}}\}$, which need to be learned. Let N_m denote the number of possible quantization values for the current pixel. The high order conditional probability consistent with all lower order constraints derived from the context and maximizing entropy is given as an exponential model[41],

$$P_{ME}[I_{m,n} = l|\mathbf{w_{m,n}}] \tag{4.1}$$
$$= \frac{e^{\left(\sum\limits_{w_{q,r}\in\mathbf{w_{m,n}}} \gamma(I_{m,n}=l, W_{q,r}=w_{q,r})\right)}}{\sum\limits_{l'=1}^{N_m} e^{\left(\sum\limits_{w_{q,r}\in\mathbf{w_{m,n}}} \gamma(I_{m,n}=l', W_{q,r}=w_{q,r})\right)}} \qquad l = 1, \ldots, N_m.$$

The detailed formulation of the IIS algorithm for learning these Lagrange multipliers is shown in chapter 1.

By using the IIS algorithm, we only need enough training data to ensure good learning of the second order joint probabilities by using frequency counts $\{N(I_{m,n} =$

$l, W_{q,r} = w_{q,r}) \quad w_{q,r} \in \mathbf{w_{m,n}}\}$. By contrast, in directly learning an unconstrained high order model, we would need enough training data to ensure good learning of the *high* order joint probabilities by using the frequency counts $N(I_{m,n} = l, \mathbf{w_{m,n}})$.

4.2.2 High order CECTCQ encoding for images

The objective of a conditional entropy-constrained encoder is to minimize the Lagrangian cost function $J \equiv D + \lambda \cdot R$ over choice of quantization indices, where D is the distortion, and R is the bit rate. For better understanding, this overall Lagrangian cost is rewritten as a sum of Lagrangian costs for each row, i.e.

$$J = \sum_{m=1}^{m=M} J_m = \sum_{m=1}^{m=M} (D_m(\underline{i_m}) + \lambda R_m(\underline{i_m}, \underline{\mathbf{w_m}})) \tag{4.2}$$

where,

$$\begin{aligned} D_m(\underline{i_m}) &= \sum_{n=1}^{n=N} d(s_{m,n}, q(i_{m,n})) \\ R_m(\underline{i_m}, \underline{\mathbf{w_m}}) &= \sum_{n=1}^{n=N} l(I_{m,n} = i_{m,n}; \mathbf{w_{m,n}}) \qquad M \geq m \geq 1. \end{aligned}$$

$l(I_{m,n} = i_{m,n}; \mathbf{w_{m,n}})$ is the VLC code length of $i_{m,n}$, which is a function of $\mathbf{w_{m,n}}$. $s_{m,n}$ is the source value at row m and column n before quantization. $q(i_{m,n})$ is the quantization value representing quantization index $i_{m,n}$. $d(s_{m,n}, q(i_{m,n}))$ is the distance function, which we choose as squared distance. $\underline{\mathbf{w_m}}$ are the high order neighborhoods ($\{\mathbf{w_{m,n}}\} \quad \forall n$) in row m. The detailed setup of our transformed coding for images can be seen in chapter 2.

In equation 4.2, we used self information $-log_2 P[I_{m,n} = i_{m,n}|\mathbf{w_{m,n}}]$ to approximate the actual VLC bit length $l(I_{m,n} = i_{m,n}; \mathbf{w_{m,n}})$. The high order *a posteriori* probability $P[I_{m,n} = i_{m,n}|\mathbf{w_{m,n}}]$ is obtained from equation 4.1. Note that this high order *a posteriori* probability reduces to $P[I_{m,n} = i_{m,n}|w_{m,n-1}, w_{m-1,n}]$ in a standard second order CECTCQ algorithm[30].

Our high order CECTCQ algorithm is a greedy approach which optimizes its solution row-by-row given the previously encoded rows. The quantization indices in each row are chosen to minimize the overall Lagragian cost J_m for that row via a TCQ algorithm[36]. TCQ has a constrained trellis structure where not all state transitions are possible. Thus, the choice of quantization indices at current state is limited by the possible transition from previous state. The metric for each transition between previous quantization choice and next quantization choice is given by $d(s_{m,n}, q(i_{m,n})) + \lambda \log_2 P[I_{m,n} = i_{m,n}|\mathbf{w_{m,n}}]$. Once the quantization indices are chosen for the current row ($\underline{i_m}$) by using our high order CECTCQ algorithm, it is being used together with the solution from previous rows ($\underline{i_{m'}} \quad m' < m$) in choosing quantization indices for the next row ($\underline{i_{m+1}}$). This process is repeated until the whole image has been passed. The pseudo code for the high order CECTCQ approach is as follows:

1. Design codebooks using the standard 2^{nd} order CECTCQ from training images.

2. Encode training images using the standard 2^{nd} order CECTCQ from the previously designed codebooks.

3. Train our high order statistical model $P[I_{m,n} = i_{m,n}|\mathbf{w_{m,n}}]$ using (4.1) from the quantization indices obtained from training images.

4. Encode test image:

   - For $m = 1$ to M,

     Use TCQ algorithm to minimize $D_m(\underline{i_m}) + \lambda R_m(\underline{i_m}, \underline{\mathbf{w_m}})$ over $\underline{i_m}$ using high order model parameters obtained from step 3.

   End

5. Performance measure on encoded test image.

Note that this codebook design process includes choosing quantization levels and its corresponding VLC codeword. Also, the formulation for the high order CECTCQ discussed can be easily changed to a standard second order CECTCQ approach[30] by reducing high order neighborhood $\mathbf{w_{m,n}}$ to $w_{m,n-1}$ and $w_{m-1,n}$.

4.3 Experiment

Our experimental setup is similar to the one in chapter 2. In our experiments, we used 23 training images and 1 test image with size of $512X512$. There are two stages in our experiments. We first design codebooks and train our high order conditioning model by using frequency counts; then test our algorithm outside the training set.

Let us consider designing codebooks and training for the high order conditioning model at this point. Each image was first linear transformed by using a 2D DCT with $8X8$ blocks. This creates 64 transformed sources sorted from the highest to lowest variance with $4096 \cdot 23$ elements in each one. We set the overall rate of 0.25 bits per symbol, achieved by only using the five sources ranked with highest variance, which were given the individual rates $\{4, 3, 3, 3, 3\}$. The rest of the source rates were set to

zero. Each source in the training set is being encoded in an iterative (4 state 2^{nd} order CECTCQ) codebook design process to reduce the overall distortion given the constraint of overall rate via the entropy-constrained TCQ version[30] of the GLA algorithm[33]. This codebook design process stops when the improvement in overall distortion falls below a specified threshold. The encoded quantization indices for these training images after the final iteration of codebook design is then used for learning our high order conditioning model (4.1) by using IIS algorithm[4]. In our experiment, each source uses a different context space/high order conditioning neighborhood, with its conditional probability specified as follows (m=1 starts from top, n=1 starts from left),

$$
\begin{aligned}
P[i_{m,n}|\mathbf{w_{m,n}}] &= P[i_{m,n}|w_{m,n-1}] \quad m=1, \\
P[i_{m,n}|\mathbf{w_{m,n}}] &= P[i_{m,n}|w_{m,n-1}, w_{m-1,n-1}, w_{m-1,n}, w_{m-1,n+1}] \quad m=2, \\
P[i_{m,n}|\mathbf{w_{m,n}}] &= P[i_{m,n}|w_{m,n-1}, w_{m-2,n-1}, w_{m-2,n}, w_{m-2,n+1}, w_{m-1,n-1}, \\
&\quad w_{m-1,n}, w_{m-1,n+1}] \quad m=3, \\
P[i_{m,n}|\mathbf{w_{m,n}}] &= P[i_{m,n}|w_{m,n-1}, w_{m-3,n-1}, w_{m-3,n}, w_{m-3,n+1}, w_{m-2,n-1}, \\
&\quad w_{m-2,n}, w_{m-2,n+1}, w_{m-1,n-1}, w_{m-1,n}, w_{m-1,n+1}] \quad m=4, \\
P[i_{m,n}|\mathbf{w_{m,n}}] &= P[i_{m,n}|w_{m,n-1}, w_{m-4,n-1}, w_{m-4,n}, w_{m-4,n+1}, w_{m-3,n-1}, \\
&\quad w_{m-3,n}, w_{m-3,n+1}, w_{m-2,n-1}, w_{m-2,n}, w_{m-2,n+1}, w_{m-1,n-1}, \\
&\quad w_{m-1,n}, w_{m-1,n+1}] \quad m=5. \qquad (4.3)
\end{aligned}
$$

From this setup, we used as high as 13th order in the 5th source. Note that each $w_{m,n}$ is a two dimensional vector $(i_{m,n}, \tilde{S}_{m,n})$.

At this point, let us consider testing the algorithm after the high order conditioning model has been learned. The test image is first DCT transformed into frequency domain. The transformed values are grouped together into different frequency sources as mentioned in chapter 2. The frequency sources/rows are sorted from highest variance to lowest (from top to bottom). For each source, we used the same rate as specified in training. We then source encode (high order CECTCQ) each row by using the codebook (quantization levels and its corresponding codewords) designed from training.

The λ value in the Lagragian cost function $D_m + \lambda R_m \quad \forall m$ is fixed for all sources for training and encoding process. We then repeated the training and encoding process with various λ values to sweep out the rate-distortion curve shown in Figure 4.1. The peak signal to noise ratio is measured after the sources have been encoded and decoded for each λ value.

4.4 Result

In Figure 4.1, we compared our high order CECTCQ approach with the standard second and third order CECTCQ approach[30]. Let us first compare the computational complexity for using our high order CECTCQ and a standard second order CECTCQ[30] to encode images. For our high order CECTCQ approach using IIS, additional number of sums are needed for estimating the high order model using second order joint probabilities. However, the computational complexity for a standard second order CECTCQ approach includes third order joint probabilities, which need roughly the same number of sums. Thus, the computational complexity for these two methods are comparable. Let us then compare the memory storage needed for using our high order CECTCQ and

a standard second order CECTCQ[30] to encode images. We define a set $I_{m,n}$ which includes all possible quantization indices for $i_{m,n}$ (i.e. $i_{m,n} \in I_{m,n}$), and a set $\tilde{\mathbf{S}}_{\mathbf{m},\mathbf{n}}$ which includes all possible Trellis states for $\tilde{S}_{m,n}$. Thus, the cardinality of the two dimensional vector $w_{q,r}$ is $N_{si} = |\tilde{\mathbf{S}}_{\mathbf{m},\mathbf{n}}| \cdot |I_{m,n}|$. Our proposed high order CECTCQ by using the IIS algorithm requires memory storage given by $N_{si} \cdot |I_{m,n}| \cdot N_f$, where N_f is the order (the number of elements in the conditioning context) of CECTCQ. The standard second order CECTCQ approach requires memory storage given by $N_{si}^2 \cdot |I_{m,n}|$. If $N_{si} \approx N_f$ (N_f is typically smaller than N_{si}), then our proposed high order method shows comparable memory storage to the lowest second order approach. In order to keep the same number of elements per each cell in 14th order joint PMF the same as the 2nd order joint PMF used in IIS by using 23 training images, the number of training images needed for direct frequency counts would be approximately 20 thousand. This is obviously not practical. By using IIS, our algorithm achieves a high order, but structurally-constrained (parameterized) PMF, consistent with a large group of lower order constraints on our model's high order PMF. Although our model is supposedly inferior to the direct learning of high order model via frequency counts given *enough* training data, our proposed algorithm overcomes the overfitting problem. It also gives a practical approach in bridging the gap of performance-complexity tradeoff of direct learning of low and high order models via frequency counts.

As seen in Figure 4.1, we were able to gain as much as approximately 1 dB in PSNR compared to the second order CECTCQ at a given rate. Note that the PSNR performance for these two methods gets closer as rate gets higher (λ becomes smaller).

This is because the rate constraint is diminishing and the mse distortion incurred by choice of quantization indices dominates, and both methods use the same codebooks.

Also, comparing the standard second and third order in this figure, we see that third order failed to outperform second order as it should. This is an example of the *overfitting* problem, where there is not enough training data (23 training images) to achieve good learning of a third order model. In fact, the performance of these two methods are fairly close in the figure. It is because we choose to replace third order conditioning probability with second order if the context for a third order probability is empty (i.e. never occurs in training set). This phenomenon does occur often (about $75 - 80$ percent of all possible third order context combinations) due to the overfitting problem.

4.5 Conclusion

From our performance comparison in Figure 4.1, our performance gain indicates a good high order modeling by using IIS in the conditional entropy constrained encoding context. Our method is not limited to high order modeling in CECTCQ. It could be applied to any high order modeling in the context of conditional entropy constrained encoding. By using IIS algorithm, we were able to significantly reduce the training set support size needed for learning a high order model in conditional entropy constrained encoding. Our proposed method makes high order entropy constrained encoding possible to implement, and definitely provides a way of bridging the gap of performance-complexity tradeoff between standard low and high order CECTCQ algorithm.

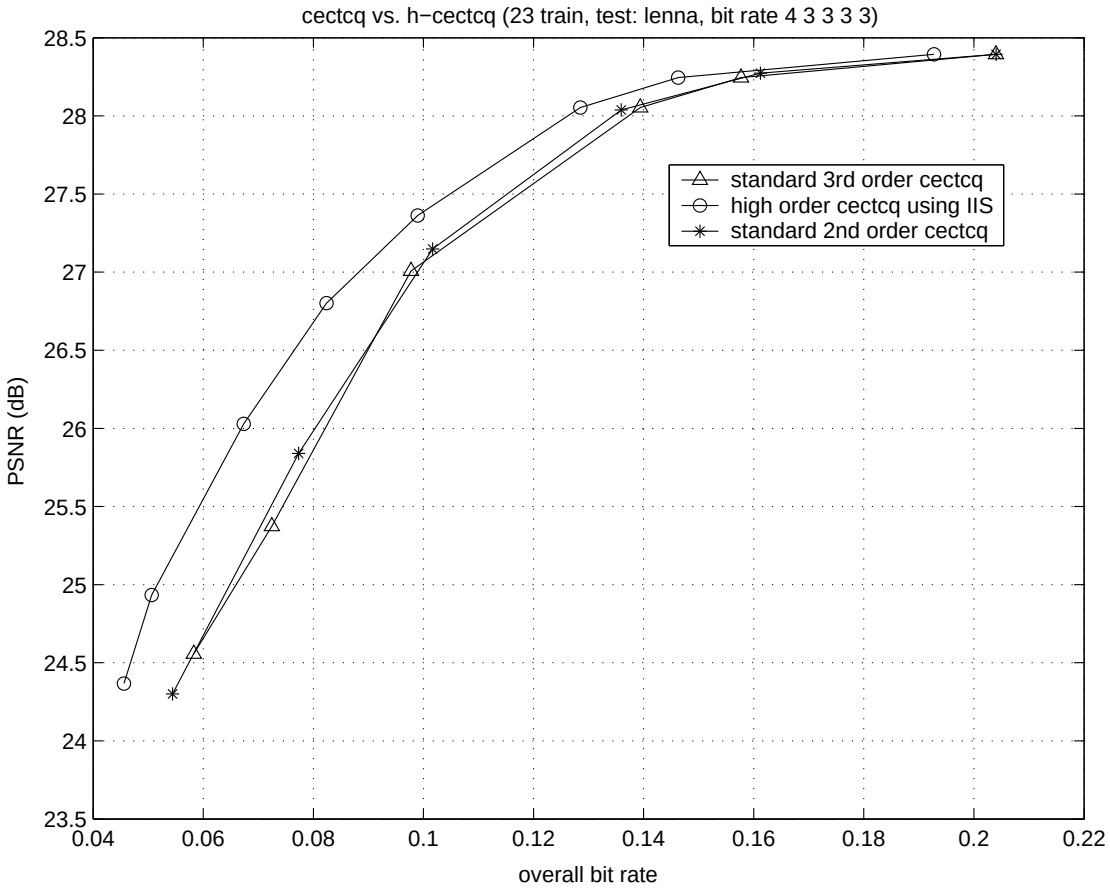

Fig. 4.1. Encoding performance (PSNR) in dB

Chapter 5

Hidden Markov Source Modeling and Extensions with Application to JSC Decoding and Lossless Compression

5.1 Introduction

In previous work on JSC decoding[40, 48, 54], Markov model[49] was always assumed for the sequence of quantization indices. By doing so, the Markovian source model at the symbol level has a limitation of finite memory, i.e. $P[I_t|I_{t-1}, I_{t-2}]$, where I_t is the random variable of transmitted quantization index at time t. Miller and Park[40] recognized the tandem of Markovian source model and noisy channel as a hidden Markov model[49] with the hidden states being the transmitted quantized indices and the observable output being noisy symbols after the channel. Villasenor et al.[19] modeled the source model as a hidden Markov model at the bit level, with the underlying states as transmitted source symbols. Their source model is approximately equivalent to a Markov model at the symbol level as in [40, 48, 54]. In [58], Turin has modeled both the source and memory channel by hidden Markov models. He modeled the source symbols as hidden Markov model at the output of the modulator after the channel coder. The hidden states of his hidden Markov model are the states of a channel coder (i.e. convolutional code). He proposed a sequence MAP(maximum *a posteriori*) decoder as an extension of the EM(expectation maximization) [3] algorithm by using the Viterbi algorithm to decode the transmitted symbols iteratively.

Due to the limitation of finite memory using a Markov model as mentioned above, a new method for modeling the source data is proposed here. This new method uses a *hidden Markov model* instead of just a Markov model[40, 48, 54] in modeling the source. Our approach is different from [58] whereas the hidden states of his hidden Markov model are the states of a channel coder (i.e. convolutional code), our hidden states do not have any underlying meaning. Also, instead of obtaining the sequence MAP solution via Viterbi algorithm[48], our JSC decoder for the *hidden Markov source model* outputs the *a posteriori* probability. This JSC decoder is a SISO (Soft Input Soft Output) decoder which is known to have superior signal to noise performance compared to a MAP decoder [40]. Our approach is also different from [40] whereas they model the source and channel tandem as a hidden Markov model, we model the source and channel tandem as a *two-layer hierarchical hidden Markov model*. In training/learning the model parameters, our new model was able to increase likelihood iteratively for the source symbol sequence by iteratively updating model parameters using the learning equations developed in this work. The JSC decoder for this new model sees the tandem of hidden Markov source model and noisy channel as a two-layer hierarchical hidden Markov model, with the first layer of hidden states being the transmitted symbols, and with the deeper layer of hidden states being the hidden states that generate source symbols, and with the observable output being the noisy channel symbols. This new model achieves "infinite memory" of a constrained form. The entropy of this hidden Markov source model was also decreased along with the likelihood increasing. The fact that the entropy is found to reduce with the increases of likelihood also indicates a possible higher compression lossless source code could be found based on this model. The performance of the first

hidden Markov source model we developed was fairly dependent of the way parameters are initialized. This motivated our second hidden Markov source model that allows the use of the learned parameters from a (standard) second order Markov source model as an initialization. In doing so, we guarantee increasing likelihood starting from this second order Markov source model. The result has shown that the signal to noise ratio of our JSC decoder for the second order hidden Markov source model is between the performance of a second and third order SAMMSE decoder based on a Markov source model. To achieve the same memory complexity, we tried our JSC decoder based on a second order hidden Markov source model and the SAMMSE decoder based on the second order Markov source model with the number of states equaling 32. When tested within training set, we were able to achieve approximately 1.1 dB gain in average signal to noise ratio compared to the SAMMSE decoder based on a second order Markov source model using an image source. Our JSC decoder based on a second order hidden Markov source model of 32 states requires memory of 67 mega bytes, while an SAMMSE decoder based on a third order Markov source model requires memory of 2.15 giga bytes! This case shows the SAMMSE decoder based on a Markov source model requires 32 times more memory compared to the JSC decoder based on a hidden Markov source model, in order to achieve a better signal to noise ratio performance. In the case of an SAMMSE decoder based on a third order Markov source model, the computational complexity was similar to our JSC decoder based on a second order hidden Markov source model, both bounded by $O(TN^4)$, while the computational complexity for an SAMMSE decoder based on the second order Markov source model is bounded by $O(TN^3)$, where T is the source symbol sequence length, and N is the number of symbols. However, the high

memory complexity of a SAMMSE decoder with third order Markov source model makes it not practical to implement.

5.2 Review of Joint Source Channel coding and SAMMSE decoder

The detailed formulation of a SAMMSE decoder[40] is shown in chapter 1, and the state diagram of a 4 state, second order hidden Markov model[49] is shown in Figure 5.1. The output of the decoder is the *a posteriori* probabilities $P[S_t = m|\underline{j}]$, where $\underline{j}$ is the received noisy symbol sequence after the noisy channel, and S_t is the hidden state at time t which refers to the possible decoded symbol variables. I_t and S_t are equivalent in this case.

In Figure 5.1, the head of the arrow shows the source symbol state where there is "a transition from" and the tail of the arrow shows the source symbol state where there is "a transition to". In each given source symbol state, there is a probability mass function associated with the noisy observed symbol, calculated using the bit error rate of the noisy channel. In this case, we are trying to find the *a posteriori* probability of each possible decoded symbol at each symbol time given the whole received sequence. The model parameters are indicated along the transition arrows. The formulation of a second and a third order SAMMSE decoder can be easily derived in the same fashion as in a first order SAMMSE decoder.

5.3 Our first hidden Markov model

Our first model is motivated by wanting to reduce the complexity while maintaining good performance by choosing the number of states to be substantially smaller

than the number of symbols, and by wanting to overcome the finite memory limitation implicit in use of the Markov source model in the previous work by Miller and Park[40], in the SAMMSE decoder. By modeling the source data by itself as a hidden Markov model, our new method is able to achieve infinite memory as will be shown in section 5.4.2. This model will require iterative parameter estimation for learning to maximize the likelihood on a training set. We have found that, in addition to increasing the likelihood, our learning reduces the entropy of the source data. This indicates our model may be useful for efficient lossless compression of the source (in addition to our primary goal of JSC decoding). In this first model, the decoder sees the system as a two layer hidden Markov model as shown in Figure 5.2. The first layer of hidden states being the transmitted symbols, and the deeper layer of the hidden states being the hidden states used to generate the source symbols. The three parameters for this model are $P[I_t|S_t, I_{t-1}]$, $P[S_t|S_{t-1}]$, $P[j_t|I_t]$. We choose to initialize $P[I_t|S_t, I_{t-1}]$, $P[S_t|S_{t-1}]$ randomly due to the fact that S_t does not have any underlying meaning that would suggest another initialization. Our results have shown that the performance in signal to noise ratio of this model is fairly sensitive to different initialization points.

In Figure 5.2, we used the same convention as in Figure 5.1. This is a two layer hierarchical hidden Markov model. At each symbol, there is a noisy channel probability involved to generate noisy symbols. Our decoder for this first model sees a direct implementation of two layer hidden Markov model. The formulation of our first hidden Markov model is skipped here because a better model will be shown to overcome its initial point dependence problem in the next section. The interested readers of our first hidden Markov source model are referred to Appendix for its detailed development.

5.4 Our new model

In the following sections, we will show the detailed formulation of learning our 2nd order hidden Markov source model parameters, and a JSC decoder based on this model. Our first order hidden Markov source model can also be derived in the same fashion, but will not be shown here for space consideration. Both models will be used for comparison, however, in the experimental result section.

Due to the performance dependence on initial points of our first model, we have developed a new hidden Markov model for the source which could use the parameters of a second order Markov source model as the initial parameter values. Our new JSC decoder sees the whole system as shown in Figure 5.3. In this new model, the model parameters are $P[S_t|S_{t-1}, S_{t-2}]$, $P[I_t|S_t, S_{t-1}]$, and $P[J_t|I_t]$. By letting $S_t = I_t$ as initial points, the parameters of our new hidden Markov source model can then be initialized to achieve equivalent Markov source model. Also, seen from the JSC decoder perspective, this new model(in Figure 5.3) at initialization becomes equivalent to the hidden Markov model used by the SAMMSE decoder in Figure 5.1. It becomes obvious that the hidden Markov model seen by the SAMMSE decoder in Figure 5.1 is a special case of the two layer hidden Markov model seen by our new JSC decoder in Figure 5.3. In the learning section, we will first show how the parameters of our new hidden Markov source model are updated iteratively by using the update equations. In the inference section, we will show how these learned model parameters can be used in our JSC decoding. Our new model is guaranteed (during training) to ascend in the likelihood starting from the second order Markov source solution. Moreover, we have seen empirically that this translates to

iterative improvement in JSC decoding performance(SNR), when our hidden Markov source model is used to model the source in a JSC decoding system. This model has also shown lower entropy through iterations of update equations compared to the second order Markov source model.

5.4.1 Learning

In this section, we will show how the two model parameters ($P[S_t = m|S_{t-1} = r, S_{t-2} = k]$, $P[I_t = k|S_t = m, S_{t-1} = r]$) are being learned via our iterative update equations. We first initialize the parameters of the hidden Markov source model by letting $S_t = I_t$ as shown in (5.1)(5.2). By doing so, the parameters of our new hidden Markov source model can be initialized to achieve equivalent second order Markov source model. We then need to calculate both the forward variable $\underline{\alpha}$ and backward variable $\underline{\beta}$ using recursive equations (5.3) and (5.4), respectively. After the forward variable $\underline{\alpha}$ and backward variable $\underline{\beta}$ are calculated, they are to be used in the update equations (5.5)(5.8) in learning the model. The same set of $\underline{\alpha}$ and $\underline{\beta}$ variables are then used to measure the likelihood(5.17) improvement for whether to stop the learning process. We also used these $\underline{\alpha}$ and $\underline{\beta}$ variables to measure the entropy(5.15) of our proposed model as an indication of how well a lossless compression would perform if an entropy coding (VLC) was applied. Our algorithm was extended from the idea of Expectation Maximization techniques developed by Baum et al.[3].

5.4.1.1 Initialization

By letting $S_t = I_t$, the parameters of our new hidden Markov source model become as follows:

$$P[S_t|S_{t-1}, S_{t-2}] = P[I_t|I_{t-1}, I_{t-2}], \tag{5.1}$$

$$P[I_t|S_t, S_{t-1}] = \delta(I_t - S_t). \tag{5.2}$$

These model parameters essentially become (at initialization) standard second order Markov model parameters, obtainable from frequency counts.

5.4.1.2 Update Equations

In this section, we first show the formulation of the forward and backward variables. Then, we will show how these variables are used in our model parameter update equations. Let i_t denotes the transmitted quantization index at time t.

The forward variable $\underline{\alpha}$ is calculated recursively from $t = 2$ to $t = T - 1$, given by

$$\alpha_t(S_t = m, S_{t-1} = r) = P[i_0, \ldots, i_t, S_t = m, S_{t-1} = r]$$

$$= \big[\sum_k \alpha_{t-1}(S_{t-1} = r, S_{t-2} = k) P[S_t = m|S_{t-1} = r, S_{t-2} = k]\big] P[i_t|S_t = m, S_{t-1} = r]. \tag{5.3}$$

$\underline{\alpha}$ is the probability of the causal part of source symbol sequence joint with the possible current and previous hidden states. α_t is calculated in a special way when $t = 0$ and $t = 1$ in the same fashion as in the standard SAMMSE decoder case[40].

The backward variable $\underline{\beta}$ is calculated recursively from $t = T-2$ to $t = 0$, given by

$$\beta_t(S_t = m, S_{t-1} = r) = P(i_{t+1}, \ldots, i_{T-1} | S_t = m, S_{t-1} = r)$$

$$= \sum_k P[S_{t+1} = k | S_t = m, S_{t-1} = r] P[i_{t+1} | S_{t+1} = k, S_t = m] \beta_{t+1}(S_{t+1} = k, S_t = m). \tag{5.4}$$

It is set to one when $t = T-1$. $\underline{\beta}$ is the probability of the anti-causal part of the source symbol sequence given the possible current and previous hidden states. β_t is calculated in a special way when $t = 0$ in the same fashion as in the standard SAMMSE decoder case[40].

We take the $\underline{\alpha}$ and $\underline{\beta}$ from (5.3) and (5.4) to be used in our learning equations (5.5) and (5.8). The set of learning equations are used to iteratively update the parameters of the model with increasing likelihood. After each iteration of the learning equation, a new $\underline{\alpha}$ variable and $\underline{\beta}$ variable from (5.3) and (5.4) need to be recalculated using these updated model parameters from (5.5) and (5.8) for the next iteration. The likelihood (5.17) also needs to be recalculated after each iteration to check the stopping criterion. The iteration is stopped when the difference of likelihood between two consecutive iterations are smaller than 0.5 and more than 50 iterations have been done.

The **first model parameter** is re-estimated by

$$P^{(n+1)}[S_t = m | S_{t-1} = r, S_{t-2} = k] = \frac{\sum_{t=0}^{T-1} P^{(n)}[S_t = m, S_{t-1} = r, S_{t-2} = k | \underline{i}]}{\sum_{t=0}^{T-1} P^{(n)}[S_{t-1} = r, S_{t-2} = k | \underline{i}]}, \tag{5.5}$$

where superscript (n) denotes the counter/index for iterations. Its numerator and denominator can be simplified to a function of the current $\underline{\alpha}$ and $\underline{\beta}$ from (5.3) and (5.4) and the model parameters from the previous iteration. The probability in the numerator is given by,

$$
\begin{aligned}
&P^{(n)}[S_t = m, S_{t-1} = r, S_{t-2} = k|\underline{i}] = \\
&\{\alpha_{t-1}(S_{t-1} = r, S_{t-2} = k)\beta_t(S_t = m, S_{t-1} = r)P^{(n)}[S_t = m|S_{t-1} = r, S_{t-2} = k]\cdot \\
&P^{(n)}[i_t|S_t = m, S_{t-1} = r]\}/ \\
&\{\sum_{m',r',k'} \alpha_{t-1}(S_{t-1} = r', S_{t-2} = k')\beta_t(S_t = m', S_{t-1} = r')\cdot \\
&P^{(n)}[S_t = m'|S_{t-1} = r', S_{t-2} = k']P^{(n)}[i_t|S_t = m', S_{t-1} = r']\}.
\end{aligned}
\tag{5.6}
$$

The probability in the denominator is given by,

$$
\begin{aligned}
&P^{(n)}[S_{t-1} = r, S_{t-2} = k|\underline{i}] = \\
&\frac{\alpha_{t-1}(S_{t-1} = r, S_{t-2} = k)\beta_{t-1}(S_{t-1} = r, S_{t-2} = k)}{\sum_{r',k'} \alpha_{t-1}(S_{t-1} = r', S_{t-2} = k')\beta_{t-1}(S_{t-1} = r', S_{t-2} = k')}.
\end{aligned}
\tag{5.7}
$$

The **second model parameter** is re-estimated by

$$
P^{(n+1)}[I_t = k|S_t = m, S_{t-1} = r] = \frac{\sum_{t=0}^{T-1} P^{(n)}[S_t = m, I_t = k, S_{t-1} = r|\underline{i}]}{\sum_{t=0}^{T-1} P^{(n)}[S_t = m, S_{t-1} = r|\underline{i}]}. \tag{5.8}
$$

Its numerator and denominator can also be simplified to a function of current $\underline{\alpha}$ and $\underline{\beta}$ from (5.3) and (5.4) and the model parameters from the previous iteration. The probability in the numerator is given by,

$$P^{(n)}[S_t = m, I_t = k, S_{t-1} = r|\underline{i}] = \\ \frac{\alpha_t(S_t = m, S_{t-1} = r)\beta_t(S_t = m, S_{t-1} = r)\delta(i_t - k)}{\sum\limits_{m',r'} \alpha_t(S_t = m', S_{t-1} = r')\beta_t(S_t = m', S_{t-1} = r')}. \tag{5.9}$$

The probability in the denominator is given by,

$$P^{(n)}[S_t = m, S_{t-1} = r|\underline{i}] = \frac{\alpha_t(S_t = m, S_{t-1} = r)\beta_t(S_t = m, S_{t-1} = r)}{\sum\limits_{m',r'} \alpha_t(S_t = m', S_{t-1} = r')\beta(S_t = m', S_{t-1} = r')}. \tag{5.10}$$

The **initial model parameters** $P[S_t = m]$ and $P[I_t = k|S_t = m]$, and $P[S_t = m|S_{t-1} = r]$ are updated in the following equations:

$$P^{(n+1)}[S_t = m] = \frac{\sum\limits_{t=0}^{T-1}\sum\limits_{r}\sum\limits_{k} P^{(n)}[S_t = m, S_{t-1} = r, S_{t-2} = k|\underline{i}]}{T}, \tag{5.11}$$

$$P^{(n+1)}[S_t = m|S_{t-1} = r] = \frac{\sum\limits_{t=0}^{T-1}\sum\limits_{k} P^{(n)}[S_t = m, S_{t-1} = r, S_{t-2} = k|\underline{i}]}{\sum\limits_{t=0}^{T-1}\sum\limits_{k} P^{(n)}[S_{t-1} = r, S_{t-2} = k|\underline{i}]}, \tag{5.12}$$

$$P^{(n+1)}[I_t = k|S_t = m] = \frac{\sum\limits_{t=0}^{T-1}\sum\limits_{r} P^{(n)}[S_t = m, I_t = k, S_{t-1} = r|\underline{i}]}{\sum\limits_{t=0}^{T-1}\sum\limits_{r} P^{(n)}[S_t = m, S_{t-1} = r|\underline{i}]}. \tag{5.13}$$

5.4.2 Entropy and Likelihood

There are two measurements for our hidden Markov source model. First is the entropy measure. Second is the likelihood measure. The source entropy is the minimum rate of lossless compression. It is therefore a good measurement of how much the source could be compressed losslessly if an actual VLC code was used. In the experimental result section, we will show that our proposed method has a lower entropy than a Markov source model with comparable memory storage requirements[40]. This is due to the fact that we are able to achieve infinite memory in a constrained form in contrast to a standard approach. This will be shown in this section. The likelihood measure is used to determine the stopping point of our learning iterations.

5.4.2.1 Entropy

The entropy of a second order Markov source model is given by

$$H\big(P[I_t = l|I_{t-1} = m, I_{t-2} = n]\big) = -\sum_m \sum_n P[I_{t-1} = m, I_{t-2} = n] \cdot \sum_l P[I_t = l|I_{t-1} = m, I_{t-2} = n] \log P[I_t = l|I_{t-1} = m, I_{t-2} = n]. \tag{5.14}$$

The entropy of our hidden Markov source model is given by

$$H\big(P[I_t = l|i_{t-1}, \ldots, i_0]\big) = -\frac{\sum_{t=0}^{T-1} \sum_l P[I_t = l|i_{t-1}, \ldots, i_0] \log P[I_t = l|i_{t-1}, \ldots, i_0]}{T}, \tag{5.15}$$

where

$$P[I_t = l|i_0, \ldots, i_{t-1}] =$$

$$\frac{\sum\limits_{m,r,k} \alpha_{t-1}(S_{t-1} = r, S_{t-2} = k) P[I_t = l|S_t = m, S_{t-1} = r] P[S_t = m|S_{t-1} = r, S_{t-2} = k]}{\sum\limits_{r,k} \alpha_{t-1}(S_{t-1} = r, S_{t-2} = k)}. \tag{5.16}$$

Equation (5.16) shows the advantage of our hidden Markov source model achieving infinite memory in a constrained form compared to the finite memory $P[I_t|I_{t-1}, I_{t-2}]$ used in the second order Markov source model[40]. It is well known in information theory that higher conditioning/memory would result in lower entropy, hence higher lossless compression. In the experimental result section, we will show our entropy advantage compared to a standard second order Markov source model.

5.4.2.2 Likelihood

The likelihood is measured after each iteration of learning equations to validate our formulation and provide a stopping criterion for our learning. This likelihood function is given by

$$\begin{aligned} P[\underline{i}|\lambda] &= \sum_{m,r} \tilde{\alpha}_t(S_t = m, S_{t-1} = r)\tilde{\beta}_t(S_t = m, S_{t-1} = r) \\ &= \sum_{m,r} \tilde{\alpha}_{T-1}(S_{T-1} = m, S_{T-2} = r), \end{aligned} \tag{5.17}$$

where $\tilde{\alpha}_t$ and $\tilde{\beta}_t$ represent the values before renormalization. The $\tilde{\alpha}_t$ and $\tilde{\beta}_t$ have to be renormalized by

$$\alpha_t(S_t = m, S_{t-1} = r) = \frac{\hat{\alpha}_t(S_t = m, S_{t-1} = r)}{\sum_{m'}\sum_{r'}\hat{\alpha}_t(S_t = m', S_{t-1} = r')}$$

$$\beta_t(S_t = m, S_{t-1} = r) = \frac{\hat{\beta}_t(S_t = m, S_{t-1} = r)}{\sum_{m'}\sum_{r'}\hat{\beta}_t(S_t = m', S_{t-1} = r')} \tag{5.18}$$

at every symbol time in equation (5.3) and (5.4) to avoid numerical underflow, where $\hat{\alpha}_t$ denotes the recursive version of renormalized $\tilde{\alpha_t}$ at symbol time t. In order to find the log likelihood of $P[\underline{i}|\lambda]$, we need to keep the scale factors used in α_t at every symbol time in order to unscale the α_t at $t = T - 1$. We have not seen anyone else being able to calculate the likelihood for source symbol sequence using the Forward/Backward algorithm in any publications. From the property of $\underline{\alpha}$ variable recursion, we realized that in order to unscale α_{T-1} we need to multiply all the scale factors $scale(t) = \sum_{m'}\sum_{r'}\hat{\alpha}_t(S_t = m', S_{t-1} = r')$ from $t = 0$ to $t = T - 1$. The likelihood then becomes

$$P[\underline{i}|\lambda] = \prod_{t=0}^{T-1} scale(t) \sum_{m,r} \alpha_{T-1}(S_{T-1} = m, S_{T-2} = r). \tag{5.19}$$

Since $\sum_{m,r} \alpha_{T-1}(S_{T-1} = m, S_{T-2} = r) = 1$ after scaling, the log likelihood is given by

$$\log(P[\underline{i}|\lambda]) = \sum_{t=0}^{T-1} \log(scale(t)), \tag{5.20}$$

where the scale factors of $\underline{\alpha}$ are obtained from (5.3).

5.4.3 Inference

In this section, a JSC decoding algorithm is developed for our new hidden Markov source model to output the *a posteriori* probabilities of the transmitted symbol sequence by using the noisy symbol sequence and the converged updated model parameters. A conditional mean estimator[40] is then used to estimate the transmitted quantization levels.

5.4.3.1 JSC decoder

A *different* set of forward and backward variables (not the same as the forward and backward variables in learning) is used here to determine the *a posteriori* probabilities. In order to make this distinction clear, we use $\underline{\check{\alpha}}$ and $\underline{\check{\beta}}$ to denote the forward and backward variables in the inference section.

The *a posteriori* probabilities are obtained by

$$\begin{aligned} P[I_t = i_t|\underline{j}] &= \frac{P[I_t = i_t, \underline{j}]}{P[\underline{j}]} \\ &= \frac{\sum_m \sum_k [\check{\alpha}_t(S_t = m, S_{t-1} = k)\check{\beta}_t(S_t = m, S_{t-1} = k)P[I_t = i|S_t = m, S_{t-1} = k]]}{\sum_m \sum_k [\check{\alpha}_t(S_t = m, S_{t-1} = k)\check{\beta}(S_t = m, S_{t-1} = k)]} \quad \forall t, \end{aligned} \tag{5.21}$$

where $\check{\alpha}_t(S_t = m, S_{t-1} = k)\check{\beta}_t(S_t = m, S_{t-1} = k) = P[S_t = m, S_{t-1} = k, \underline{j}]$.

The $\check{\alpha}_t$ variable is calculated recursively by

$$\check{\alpha}_t(S_t = m, S_{t-1} = k) = P[j_0, \ldots, j_t, S_t = m, S_{t-1} = k]$$

$$= \sum_q \sum_l \check{\alpha}_{t-1}(S_{t-1} = k, S_{t-2} = l) P[S_t = m | S_{t-1} = k, S_{t-2} = l] \cdot \tag{5.22}$$

$$P[I_t = q | S_t = m, S_t - 1 = k] P[j_t | I_t = q]$$

from $t = 2$ to $t = T - 1$. $\check{\alpha}_t$ is the probability of the causal part of the received symbol sequence joint with the possible current and previous hidden states. $\check{\alpha}_t$ is calculated in a special way when $t = 0$ and $t = 1$ in the same fashion as in a standard SAMMSE decoder[40].

The $\check{\beta}_t$ variable is calculated recursively by

$$\check{\beta}_t(S_t = m, S_{t-1} = k) = P[j_{t+1}, \ldots, j_{T-1} | S_t = m, S_{t-1} = k]$$

$$= [\sum_q \sum_l P[S_{t+1} = l | S_t = m, S_{t-1} = k] P[I_{t+1} = q | S_{t+1} = l, S_t = m] \cdot \tag{5.23}$$

$$P[j_{t+1} | I_{t+1} = q] \check{\beta}_{t+1}(S_{t+1} = l, S_t = m)]$$

from $t = T - 2$ to $t = 1$. It is set to one when $t = T - 1$. $\check{\beta}_t$ is the probability of the anti-causal part of the received symbol sequence given the possible current and previous hidden states. $\check{\beta}_t$ is calculated in a special way when $t = 0$ in the same fashion as in a standard SAMMSE decoder[40].

5.4.3.2 Performance Measure for JSC decoder

We used two performance measures for the JSC decoder. Let $X(t)$ denotes the original value before quantization at time t, and $\hat{X}(t)$ denotes the reconstructed value after conditional mean estimator[40] at time t. The first performance measure is the average signal to noise ratio given by $\text{Average SNR} = 10\log_{10}(\frac{\sum_{t=0}^{T-1} X(t)^2}{\text{Average MSE}})$, where $\text{MSE} = \sum_{t=0}^{T-1}(X(t) - \hat{X}(t))^2$. The second performance measure is the average peak signal to noise ratio commonly used on image decoding given by $\text{Average PSNR} = 10\log_{10}(\frac{255^2}{\text{Average MSE}})$, where the average mse is obtained by taking the average value of mse for a predetermined number of channel realizations. This is to make sure that our measurement is close to the expected value.

5.5 Experimental Description, Results

Let us first consider testing within the training set at this point. This means we used the same set of sample data for training the statistical model and transmitting over noisy channel to create noisy symbol sequence. All the JSC decoding performance (ASNR) were averaged over 5 channel realizations to approximate its expected values.

In Table 5.1, we are showing comparisons of JSC decoders and source entropy based on 1) our first 2nd order hidden Markov source model with 20 random initializations for model parameters, 2) our new 2nd order hidden Markov source model with 20 random initializations for model parameters, 3) our new 2nd order hidden Markov source model with 2nd order Markov source model initialization, and 4) a standard 2nd

order Markov source model (SAMMSE)[40]. In this part of experiment, we used Gauss-Markov source with the correlation coefficient chosen to be 0.9, and the source symbol length chosen to be 4096. We used a uniform scalar quantizer. We also chose the number of symbols and number of states both equal to 16, and the bit error rate to be 0.05 for the assumed binary symmetric channel. From this table, we can see the encoder and decoder performance of both our first and new hidden Markov source model with random initializations are fairly dependent on initial points. The performance dependency on random initializations is our motivation for the new model, which could be initialized with a Markov source model. Although the decoding performance of our new model with random initializations are better than our first model with random initializations 85% of the time in this experiment, they are not always better than a standard 2nd order SAMMSE decoder (uses a 2nd order Markov source model)[40]. The decoder performance gain of the JSC decoder using our new 2nd order model initialized with a 2nd order Markov source model compared to a 2nd order SAMMSE decoder[40] is small in this case. However, we will show a more significant gain in the following experiments. Also, we note that our 2nd order new model with random initializations does show a lower rate in entropy. This indicates a higher compression could be achieved with a VLC encoder.

In Table 5.2, we are showing comparisons of JSC decoders and source entropy based on 1) our new 2nd order hidden Markov source model with 2nd order Markov source model initialization and 2) a standard 2nd order Markov source model (SAMMSE decoder)[40]. In this experiment, a Gauss-Markov source is used with 32 symbols (bit rate = 5). The correlation coefficient was chosen to be 0.9. A uniform scalar quantizer

was used to generate the quantized symbol sequence. The bit error rate of the binary symmetric channel was chosen to be 0.05, and the length of data sequence was 4096 symbols. The conditional probabilities between symbols were obtained by using frequency counts from the source symbols, which is the maximum likelihood estimator for a Markov source model. The parameters of the new hidden Markov source model were updated iteratively from the second order conditional probabilities until the likelihood of the data converged after 50 iterations and the difference between two consecutive likelihoods was less than 0.5. The result show that the entropy dropped about 0.5 bits per symbol while the signal to noise ratio gained about 1.7 dB by comparing the JSC decoder using our 2nd order hidden Markov source model with the 2nd order SAMMSE decoder[40]. This performance gain is significant. The computational complexity for the 2nd order SAMMSE decoder[40] is bounded by $O(N^3T)$, while our JSC decoder using the new 2nd order hidden Markov source model is bounded by $O(N^4T)$. Although the JSC decoder for our new 2nd order hidden Markov source model is computationally more complex than the 2nd order SAMMSE decoder[40], the memory storage complexities are similar.

We further tested our algorithm using a gray scale image as source. In this case, we used a 512x512 pixel Lenna image as source. The image is first divided into 8x8 blocks, then a 2D DCT transformation is applied to each block. The detailed formulation for transforming the image was shown in chapter 2. We assigned the bit rates of {3, 2, 1, 1, 1} to five frequency coefficients with highest variance to achieve a target bit rate of 0.125 bits per symbol, while the rest of the frequency coefficients were set to 0 bits. A four state trellis coded quantizer was used for source encoding. The parameters of our new 2nd order hidden Markov source model were first evaluated using the frequency counts

by letting $S_t = I_t$. It was then trained until the stopping criterion was satisfied using the update equations in the learning section. The bit error rate of binary symmetric channel was chosen to be 0.05. The increase of likelihood and decrease of entropy is shown in Figure 5.4 and Figure 5.5 respectively for the DC coefficient (highest variance). The *a posteriori* probabilities produced by a JSC decoder are then used in the conditional mean estimator[40] for estimating the transmitted frequency coefficient symbols. After all the frequency symbols were estimated, the inverse DCT transformation was used to obtain the estimation of the original image in the spatial domain. Even though our computational complexity is higher, the JSC decoder using our new second order hidden Markov source model has similar memory storage complexity compared to the standard second order SAMMSE decoder[40]. By using our new second order hidden Markov source model, we were able to gain approximately 1.1 dB in average PSNR (peak signal to noise ratio) in decoding and reduce approximately 0.38 bits per symbol in entropy. The standard third order SAMMSE decoder was also compared. It has similar computational complexity compared to our JSC decoder using the new second order hidden Markov source model, but the decoder memory requires 2.15 giga byte while our new model only requires 67 mega byte. This memory requirement for the standard third order SAMMSE decoder was at the limitation of modern computers, which makes its implementation not practical. The entropy and average PSNR performance comparisons are shown in Table 5.3.

Let us consider testing outside the training set at this point. This means we used a larger set of train data for training the statistical model, and a different set of test data for transmitting over the noisy channel to create the noisy channel symbol sequence. The

model parameters trained using a training set is therefore assumed known at the decoder. Thus, the transmission of large side information on model parameters is avoided. In all the following experiments: 1) we used a standard Markov source model for initializing our new hidden Markov source model, 2) the same set of 10 training images and lenna test image were used in all following experiments where image source is used, 3) the conditional probabilities between symbols were obtained by using frequency counts from the training source symbols, 4) the parameters of the new hidden Markov source model were updated iteratively from these conditional probabilities until the likelihood of the data converged after 50 iterations and the difference between two consecutive likelihoods was less than 0.5. Although we demonstrated consistent decoding performance gain when testing within the training set, we found this performance is inconsistent when testing outside training set. This is due to the overfitting problem. However, the merit of our mathematical developments and possible contributions/extensions to other applications should not be overlooked. Several experiments are included as follows:

Experiment 1):

In this experiment, a Gauss-Markov source is used with the correlation coefficient chosen to be 0.9. A uniform scalar quantizer with 16 symbols(bit rate=4) is used to generate the quantized symbol sequence. The bit error rate of the binary symmetric channel is chosen to be 0.05. The length of training data sequence is 40960 symbols, and the length of testing data sequence is 4096 symbols. The decoding performance comparison is shown in Table 5.4. As shown in this table, we can see that when testing outside the training set our 2nd order model has a decoder performance loss of about 0.15 dB compared to a 2nd order SAMMSE decoder. However, when training using the test set

our method has a decoder performance gain of about 0.27 dB compared to a 2nd order SAMMSE decoder. This is an example of overfitting problem, where our training set performance is not consistent with test set. The overfitting problem occurs when the model parameters trained by using the training set is not a good representation of the test set. This problem could be overcome by having larger training data set. This also means longer time needed for training our model parameters.

Experiment 2):

In this experiment, an AC1(second highest variance) DCT transformed image source is used. A 4 state TCQ source encoder with bit rate equal to 2 is used to generate the quantized symbol sequence. The bit error rate of the binary symmetric channel is chosen to be 0.05. The length of training data sequence is 40960 symbols (from 10 images), and the length of testing data sequence is 4096 symbols (from 1 image). The decoding performance comparison is shown in Table 5.5. This is another example of overfitting problem, whereas the JSC decoder of our 2nd order model has a performance loss of about 0.28 dB when testing outside the training set, we have a performance gain of about 0.6 dB compared to a 2nd order SAMMSE decoder when training using the test set.

Experiment 3):

In this experiment, a DC(highest variance) DCT transformed image source is used. A DPCM source encoder with predictor coefficient equal 0.9 and bit rate equal to 3 and 5 are used to generate the quantized symbol sequence. The bit error rate of the binary symmetric channel is chosen to be 0.05. The length of training data sequence is 40960 symbols (from 10 images), and the length of testing data sequence is 4096 symbols

(from 1 image). The decoding performance comparison is shown in Table 5.6. In this experiment, the results are obtained from testing outside the training set. When the bit rate is 5, our new 1st order model shows a decoder performance gain of 1.8 dB compared to a 1st order SAMMSE decoder. When the bit rate is 3, our new 1st order model shows a decoder performance similar to a 1st order SAMMSE decoder. This is because when a data sequence has higher variance (bit rate = 5), our more powerful model(achieves infinite memory in likelihood) is able to capture higher degree of statistical dependency given no overfitting problem. A less powerful model (a 2nd order SAMMSE decoder) may show comparable result to our model if data variance is lower (bit rate = 3). In this experiment, the 1st order models do not have overfitting problem due to less training data needed to minimize the model cost.

Experiment 4):

In this experiment, a DC(highest variance) DCT transformed image source is used. A DPCM source encoder with predictor coefficient equal 0.5 and bit rate equal to 3 is used to generate the quantized symbol sequence. The bit error rate of the binary symmetric channel is chosen to be 0.05. The length of training data sequence is 40960 symbols (from 10 images), and the length of testing data sequence is 4096 symbols (from 1 image). The decoding performance is measured after JSC and DPCM decoder. Its comparison is shown in Table 5.7. The JSC decoder of our 2nd order model has a performance loss of about 1 dB compared to a second order SAMMSE decoder when testing outside the training set. This is another case where overfitting problem occurs.

Experiment 5):

In this experiment, an DC(highest variance) DCT transformed image source is used.

A DPCM source encoder with specified predictor coefficient(ρ) and bit rate equal to 4 is used to generate the quantized symbol sequence. The bit error rate of the binary symmetric channel is chosen to be 0.05. The length of training data sequence is 40960 symbols (from 10 images), and the length of testing data sequence is 4096 symbols (from 1 image). The decoding performance is measured after JSC and DPCM decoder. Its comparison is shown in Table 5.8. When $\rho = 0.9$, the JSC decoder using our new 2nd order model has a performance gain of about 1.2 dB compared to a 2nd order SAMMSE decoder. When $\rho = 0.75$, the JSC decoder using our new 2nd order model has a performance gain of about 0.33 dB compared to a 2nd order SAMMSE decoder. Although we have more decoding performance gain at $\rho = 0.9$, both our method and the 2nd order SAMMSE decoder have better decoding performance when $\rho = 0.75$. This indicates there is more residual redundancy produced at the output of the DPCM encoder when $\rho = 0.75$, and both decoders were able to use this to achieve better decoding performance. However, when the residual redundancy is not as high ($\rho = 0.9$), our proposed method is able to capture more higher order residual redundancy and show a much better decoding performance gain compared to a 2nd order SAMMSE decoder.

5.6 Conclusion

In this work, we proposed a novel technique to bridge the gap of decoding performance between a standard second and third order SAMMSE decoder[40]. Although the computational complexity of our method is roughly equivalent to a standard third order SAMMSE decoder, our memory complexity is much smaller. The memory complexity of

a standard third order SAMMSE decoder is almost at the maximum capacity of modern computers. Our proposed method is also able to reduce entropy, which translates to higher lossless compression. In the experiments where we test within training set, we saw a consistent decoding performance gain of our proposed method compared to a SAMMSE decoder with the same order. However, this performance gain wasn't consistent when testing outside the training set. This was due to the overfitting problem. There were also cases where we gain decoding performance when testing outside the training set. Therefore, in this application, we may need to be selective on what sources to apply our proposed model on if model cost could be measured to avoid overfitting problem. Other possible extensions/applications need to be investigated using our proposed approach.

	our first 2nd order hidden Markov source model(20 random init.)	our new 2nd order hidden Markov source model (20 random init.)	our new 2nd order hidden Markov source model (2nd order Markov source model init.)	2nd order Markov source model (SAMMSE decoder)
Entropy	2.138605-2.183193 bits/symbol	1.754702-1.801514 bits/symbol	2.032780 bits/symbol	2.109267 bits/symbol
Average Signal to Noise Ratio	12.668941-13.042831 dB	12.690612-13.237833 dB	13.185411 dB	13.016163 dB

Table 5.1. Performance comparisons of JSC decoders and source entropy based on different source models using a Gauss-Markov source with correlation coefficient=0.9. Bit rate=4

	our 2nd order hidden Markov source model	2nd order Markov source model (SAMMSE decoder)
Entropy	2.165382 bits/symbol	2.677182 bits/symbol
Average Signal to Noise Ratio	16.919468 dB	15.270254 dB

Table 5.2. Performance comparisons of JSC decoders and source entropy based on different source models using a Gauss-Markov source with correlation coefficient=0.9. Bit rate=5

	JSC decoder using our new 2nd order model	2nd order SAMMSE	3rd order SAMMSE
average PSNR	24.076560 dB	22.947938 dB	24.402806 dB
memory requirement	approx. 67 MB	approx. 67 MB	approx. 2.15 GB
computational complexity	$O(N^4T)$	$O(N^3T)$	$O(N^4T)$

Table 5.3. Performance comparisons of JSC decoders and source entropy based on different source models using a gray scale 512x512 Lenna image as source. N is the number of symbols. T is the length of the symbol sequence.

	JSC decoder using our new 2nd order model	2nd order SAMMSE
average SNR (test outside train set)	12.834730 dB	12.976038 dB
average SNR (train using test set)	13.526475 dB	13.252573 dB

Table 5.4. Performance comparison using GM source. Uniform scalar quantizer. 40960 training samples. 4096 testing samples

	JSC decoder using our new 2nd order model	2nd order SAMMSE decoder
average SNR (test outside train set)	2.724226 dB	3.002464 dB
average SNR (train using test set)	4.295244 dB	3.639532 dB

Table 5.5. Performance comparison using a AC1 DCT transformed source with TCQ encoder. 40960 training samples. 4096 testing samples.

	JSC decoder using our new 1st order model	1st order SAMMSE decoder
average SNR (bit rate=5)	12.988827 dB	11.194052 dB
average SNR (bit rate=3)	10.865729 dB	10.865978 dB

Table 5.6. Performance comparison using a DC DCT transformed source with DPCM encoder. Test outside the training set. 40960 training samples. 4096 testing samples.

	JSC decoder using our new 2nd order model	2nd order SAMMSE decoder
average SNR	15.117839 dB	16.139531 dB

Table 5.7. Performance comparison using a DC DCT transformed source with DPCM encoder. Test outside the training set. 40960 training samples. 4096 testing samples.

	JSC decoder using our new 2nd order model	2nd order SAMMSE decoder
average SNR ($\rho = 0.9$)	3.10555 dB	1.921901 dB
average SNR ($\rho = 0.75$)	3.813224 dB	3.482452 dB

Table 5.8. Performance comparison using a DC DCT transformed source with DPCM encoder. Test outside the training set. 40960 training samples. 4096 testing samples.

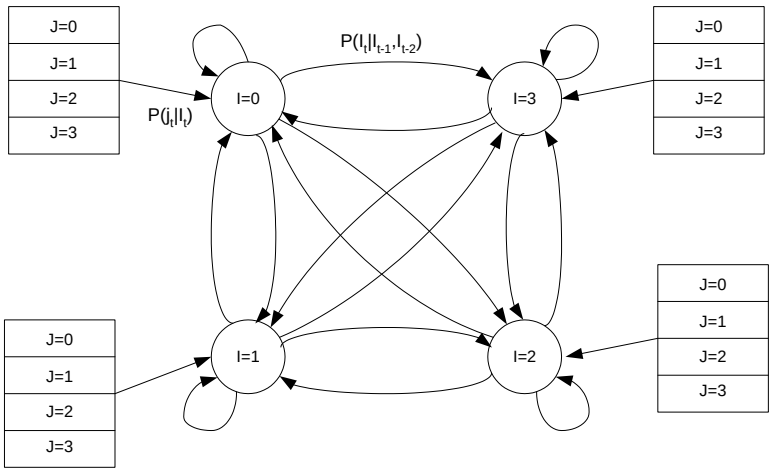

Fig. 5.1. The 2nd order hidden Markov model seen by SAMMSE decoder

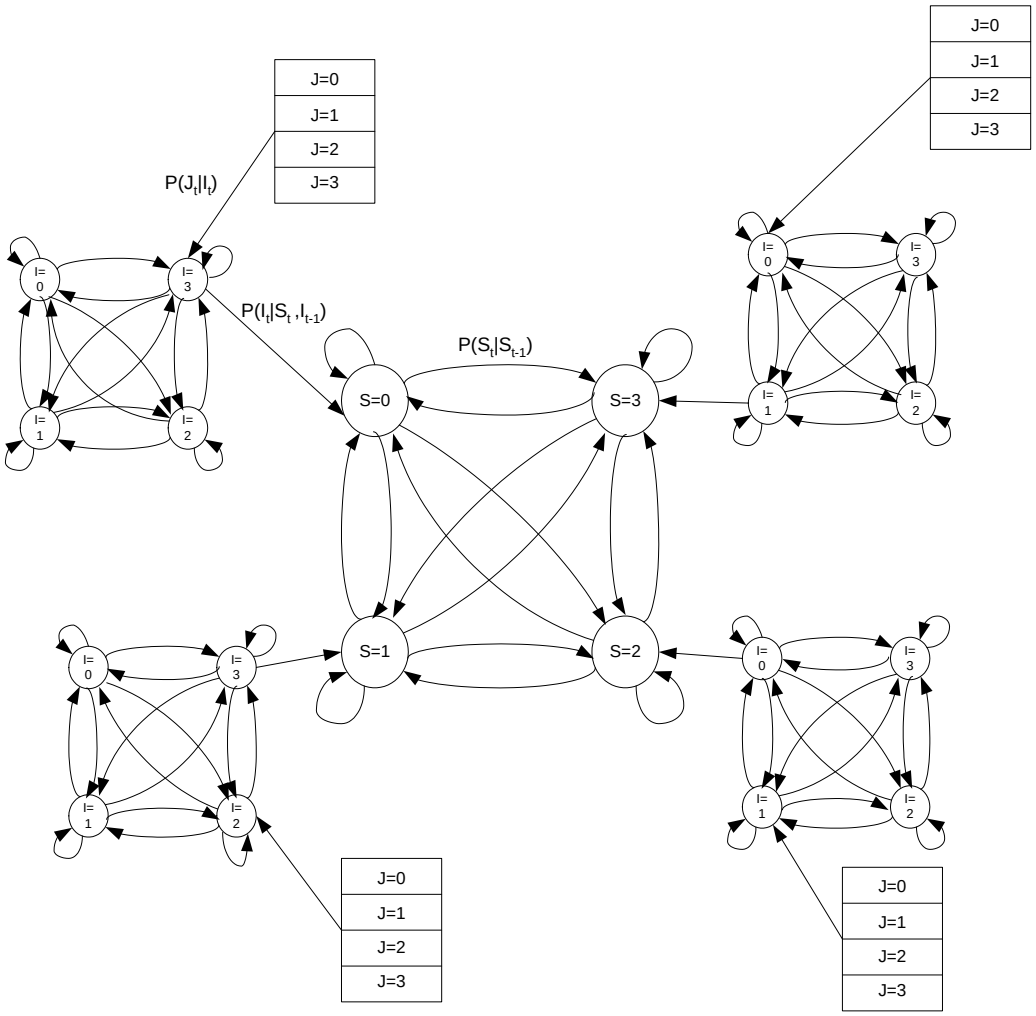

Fig. 5.2. Our first model as seen by the decoder

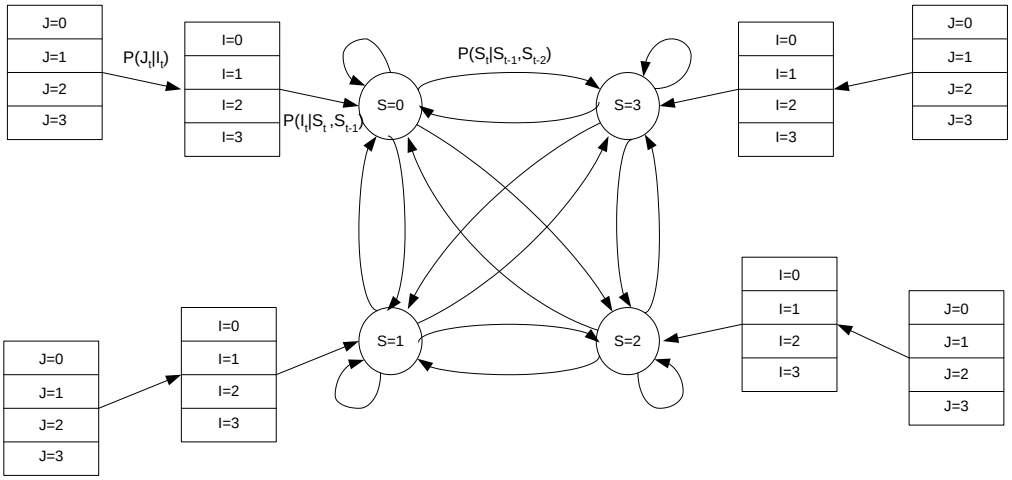

Fig. 5.3. Our new model seen by the decoder

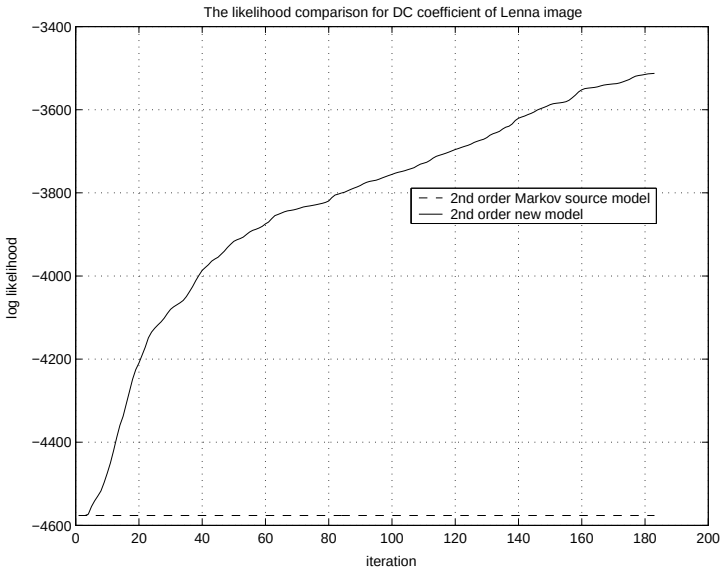

Fig. 5.4. Likelihood comparison using DC coefficient of Lenna image

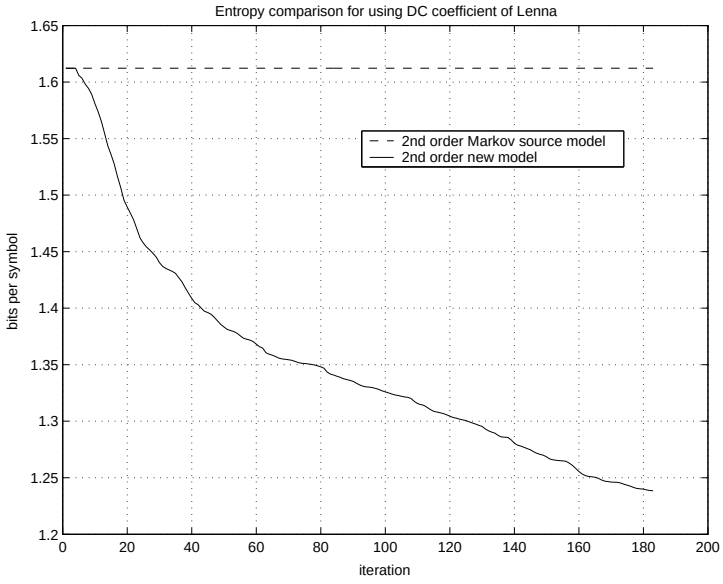

Fig. 5.5. Entropy comparison using DC coefficient of Lenna image

Chapter 6

Joint Source-Channel Image Decoding via a Sequence-based Extension of Mean-Field Techniques

6.1 Abstract

JSC decoding techniques can provide significant error correction capability without any use of explicit channel coding. However, when applied to decoding images, there are often high order (two-dimensional) dependencies that should be modeled if one is to obtain the maximum JSC decoding benefit. Unfortunately, 2D detection and estimation are quite challenging optimization problems. Existing JSC methods are tailored for 1D sources and, thus, for capturing low order dependencies. These methods, when applied to higher order models, are often quite heuristic and suboptimal. Thus, we propose new decoding techniques based on mean-field theory. An improved mean-field approximation, specifically suited for labeling problems on image support, was recently proposed by Miller et al. and applied to image segmentation [42]. Here, this new method (as well as basic mean-field annealing) is both applied and extended to address JSC image decoding. We report gains of 1-2 dB in PSNR over sequence maximum *a posteriori*, sequence approximate MMSE, and other JSC decoding methods for a basic image compression setting communicating over a binary symmetric channel.

6.2 Introduction

Several JSC decoding techniques have been previously proposed, e.g. [40, 48, 54]. These methods either treat the decoding problem as one of *hard* 0-1 symbol/sequence detection, tackled by dynamic programming [48], [54], or one of *soft* minimum mean-squared estimation (MMSE) [40]. The latter approach was based on a hidden Markov model and the Forward/Backward (F/B) algorithm [3]. While the aforementioned techniques were all proposed specifically for 1D sources, i.e. time series, they have also been applied to sources with 2D support, i.e. images, by e.g. forming 1D sources from the rows of the image. However, when the statistical model for the encoded image expresses 2D dependencies, detection and estimation are both challenging optimization tasks. 1D techniques applied in this setting are quite heuristic and may lead to poor, suboptimal solutions. Other (non-sequence-based) greedy techniques, e.g. [14], are likewise suboptimal.

The problem of JSC image decoding falls within a *family* of discrete-parameter cost minimization problems defined on image support. These problems include decoding, entropy-constrained image encoding [7], halftoning, as well as image segmentation [42]. In each case, assigning indices/labels to the pixel sites determines the value of a global objective function, one intrinsic to the problem at hand. For these problems, finding the globally optimal solution is only ensured by an (utterly infeasible) exhaustive search. Greedy optimization strategies are practically implementable, but typically quite suboptimal. Recently, several authors have tackled this general image-based problem, attempting to improve upon existing solutions. [7] proposed an iterative hillclimbing

algorithm which uses dynamic programming as a local optimization "step", repeatedly applied to the rows (or columns) of the image. Each image pass is guaranteed to improve the global cost function, or in the worst case to leave it unchanged. [11] proposed a soft input/soft output iterative optimization, alternately applied along the image rows and columns, based on "turbo decoding" concepts. One advantage of [7] is the theoretically guaranteed improvement with each image pass. No such guarantees are provided for the method in [11] – in fact, it is unclear whether this method descends in any objective function. However, a potential advantage of [11] is its propagation of soft decoding information. By contrast, the method in [7] is restricted to optimizing within the space of hard 0-1 index assignments. This feature may make this (descent) method more sensitive to the choice of initialization, and thus to finding poor local optima.

Recently, [42] proposed an image-based assignment technique based on an extension of mean-field annealing [5],[38] that preserves attractive features from both of the aforementioned methods. Similar to [11], the method in [42] propagates "soft" information – this is effected by optimizing over the (larger) space of *probability mass functions* (pmfs) on index/label assignments, rather than optimizing directly over the "hard" assignments. Moreover, similar to the "turbo decoding" in [11], the method from [42] is iterative, with each pass over the image revising the probabilistic assignments. However, the method from [42] also differs from [11] in two important respects. First, the method is formally derived via the maximum entropy principle and can be seen as an extension of techniques from mean-field theory in statistical physics. Second, unlike [11] the method from [42] descends in a (special) cost function – the Lagrangian/free energy associated with the maximum entropy problem under consideration. In [42], this new mean-field

technique was formulated specifically for image segmentation. Here, we both apply and extend this method to address JSC decoding. Applications to other image-based problems may be considered in future work. In section 3, we introduce mathematical notation and the basic problem statement. In section 4, we develop a basic mean-field theory approach for JSC decoding. In section 5, we sketch development of a mean-field extension, based on the work in [42], for the decoding problem. Finally, section 6 reports on an experimental comparison of a variety of JSC decoding methods.

6.3 Problem Statement

For clarity's sake, but without loss of generality, assume (simple) spatial-domain scalar quantization of the image gray levels. Denote the quantization index set by $\mathcal{L} = \{1, 2, \ldots, L\}$, with the transmitted index random variable at (x, y) denoted $I(x, y)$, its realization denoted $i(x, y)$, and with the array for the entire image denoted $\mathbf{I} = \{I(x, y)\ \forall x, y\}$. For concreteness, suppose the quantized image is modeled using a second order (causal), spatially invariant Markov model[1], i.e. based on the conditional probabilities $\{P[i(x, y)|i(x-1, y); i(x, y+1)]\}$[2]. Assume each index $I(x, y)$ is transmitted over a binary symmetric channel, with the received, corrupted index denoted $j(x, y)$ and with the corrupted array given by $\mathbf{j} = \{j(x, y)\ \forall x, y\}$. Then, the joint likelihood

[1] The decoding techniques developed here are also consistent with more complex models.

[2] This requires specialization for the leftmost column and the topmost row.

function for the transmitted and received index arrays is given by

$$\begin{aligned} L(\mathbf{i}) \equiv \log P[\mathbf{i};\mathbf{j}] &= \sum_{(x,y)} (\log P[i(x,y)|i(x-1,y);i(x,y+1)] \\ &+ \log P[j(x,y)|i(x,y)]). \end{aligned} \tag{6.1}$$

A desired decoding objective is to maximize $L(\mathbf{i})$ over the set of possible decoded arrays $\{\mathbf{i}\}$. Even for this simple, second-order image model, the only method guaranteeing determination of the optimal solution $\mathbf{i}^* = \arg\max_{\mathbf{i}} L(\mathbf{i})$ is an (utterly infeasible) exhaustive search over all possible image encodings. While greedy techniques [14] and 1D optimization techniques [48],[40] are practically implementable alternatives, these approaches are typically quite suboptimal. Interestingly, mean-field techniques, e.g. [5], [38] have not been previously attempted for this problem. We will thus propose both a basic (standard) mean-field technique and a method based on the mean-field extension recently proposed in [42].

6.4 Basic Mean-Field Approach

Although mean-field techniques can be formulated in several different ways, we consider a framework based on the *maximum entropy principle*, e.g. [26]. Consider the joint probability mass function (pmf) $P[\mathbf{I} = \mathbf{i}|\mathbf{J} = \mathbf{j}]$. One approach to the decoding problem is to seek an accurate estimate for this joint pmf. Given the joint pmf, one can then marginalize to obtain *a posteriori* probabilities for each pixel site. These in turn can be used as the basis for a maximum *a posteriori* detection rule or for a conditional mean (MMSE) estimator applied at each pixel. A basic mean-field approach for this joint pmf

estimation objective can be developed by first assuming the indices at each of the pixel sites are conditionally independent, i.e. $P[\mathbf{I}=\mathbf{i}|\mathbf{J}=\mathbf{j}] = \prod_{(x,y)} P[I(x,y)=i(x,y)|\mathbf{j}]$. In this case, the Shannon joint entropy is the sum of *site-wise* entropies, i.e. $H = -\sum_{(x,y)}(\sum_{k\in\mathcal{L}} P[I(x,y)=k|\mathbf{j}]\log P[I(x,y)=k|\mathbf{j}])$. Also, the average of the negative joint log likelihood with respect to the joint pmf $P[\mathbf{I}=\mathbf{i}|\mathbf{J}=\mathbf{j}]$ is

$$
\begin{aligned}
&< U > \equiv -E[L(\mathbf{i})] = -\sum_{(x,y)}\sum_{k\in\mathcal{L}} P[I(x,y)=k|\mathbf{j}](\log P[j(x,y)|i(x,y)] + \\
&\sum_m\sum_n P[I(x-1,y)=m|\mathbf{j}]P[I(x,y+1)=n|\mathbf{j}]\cdot \\
&\log P[I(x,y)=k|I(x-1,y)=m; I(x,y+1)=n]).
\end{aligned}
$$

In a basic mean-field annealing (MFA) approach e.g. [26],[38], one chooses the pmfs $\{P[I(x,y)|\mathbf{j}]\}$ to minimize the free energy $F =< U > -TH$, subject to constraints ensuring each pmf is valid, i.e. sums to one. The fixed point iterations of a site pmf is:

$$
\begin{aligned}
&P^{(t)}[I(x,y)=k|\mathbf{j}] = \mathrm{EXP}\big(\frac{1}{T}(\log P[j(x,y)|I(x,y)=k] \\
&+\sum_{m=1}^{L}\sum_{n=1}^{L} P^{(t)}[I(x-1,y)=m|\mathbf{j}]P^{(t)}[I(x,y+1)=n|\mathbf{j}] \\
&\cdot\log P[I(x,y)=k|I(x-1,y)=m, I(x,y+1)=n] \\
&+\sum_{m=1}^{L}\sum_{n=1}^{L} P^{(t-1)}[I(x+1,y)=m|\mathbf{j}]P^{(t)}[I(x+1,y+1)=n|\mathbf{j}] \\
&\cdot\log P[I(x+1,y)=m|I(x,y)=k, I(x+1,y+1)=n] \\
&+\sum_{m=1}^{L}\sum_{n=1}^{L} P^{(t-1)}[I(x,y-1)=m|\mathbf{j}]P^{(t-1)}[I(x-1,y-1)=n|\mathbf{j}] \\
&\cdot\log P[I(x,y-1)=m|I(x-1,y-1)=n, I(x,y)=k])\big)
\end{aligned}
\tag{6.2}
$$

Where (t) is the counter/index for iterations. Here, T is a Lagrange multiplier that acts as a "temperature", determining the level of entropy in the solution. The minimization can be performed at fixed T, e.g. $T = 1$ [3]. Alternatively, a computational *annealing* can be performed, where T is varied from high to low values, with the converged solution tracked from one T value to the next. Annealing provides some potential for avoiding local optima, albeit at the cost of a significant increase in computational complexity. At any T, the minimization can be performed via *fixed point iterations* (FPIs) (6.2). Serial application of the FPIs (one pixel at a time) achieves a guaranteed descent in the free energy [26],[42]. Parallel (simultaneous) pixel updating can also be used [42]. However, this approach is not guaranteed to descend in F. Once optimized, the *a posteriori* probabilities $\{P[I(x,y) = m|\mathbf{j}]\}$ can be used either for maximum *a posteriori* detection of the quantization indices, or for (more direct) MMSE estimation of the image gray levels.

6.5 Sequence-Based Extension of Mean-Field Annealing

While basic MFA optimizes over the space of joint pmfs $P[\mathbf{I} = \mathbf{i}|\mathbf{j}]$, the optimization is restricted to those pmfs expressing conditional independence of the indices, given the observed array $\mathbf{j}$. If the optimization were instead over a *less* restrictive class of pmfs, a better solution might be found. Consider the random *sequence* of indices for row y, i.e. $\underline{I_y} \equiv (I(y,1), I(y,2), \ldots, I(y,N)) \in \mathcal{L}^N$, assuming an $N \times N$ image. Suppose that, rather than assuming individual pixel indices $I(x,y)$ are conditionally independent, we

[3] In this case, the resulting method can be related to the well-known Expectation/Maximization algorithm.

assume the index *sequences* associated with distinct rows are conditionally independent, given $\mathbf{J} = \mathbf{j}$. In this case, the joint pmf factors as $P[\mathbf{I}|\mathbf{j}] = \prod_{y=1}^{N} P[\underline{I_y}|\mathbf{j}]$. With this factorization, indices for pixels in different rows are (still) conditionally independent. However, dependencies are preserved for indices within the same row. In [42], an MFA extension was proposed for optimizing over this less restrictive class of joint pmfs. This method was applied to image segmentation. Here, we next sketch an application and extension of this generalized MFA approach in order to address JSC decoding.

6.5.1 Formulation

With the assumption of conditionally independent row vectors, Shannon's joint entropy is now $H = -\sum_{y=1}^{N} \sum_{\underline{i} \in \mathcal{L}^N} P[\underline{I_y} = \underline{i}|\mathbf{j}] \log P[\underline{I_y} = \underline{i}|\mathbf{j}]$. Also, the average negative log-likelihood is

$$\begin{aligned} &< U > \equiv -E[L(\mathbf{i})] \\ &= -\sum_{y=1}^{N} \sum_{\mathbf{i} \in \mathcal{L}^N} \sum_{\mathbf{i}' \in \mathcal{L}^n} P[\underline{I_y} = \underline{i}|\mathbf{j}] P[\underline{I_{y+1}} = \underline{i'}|\mathbf{j}] U_y(\underline{i}, \underline{i'}), \end{aligned}$$

where

$$U_y(\underline{i}, \underline{i'}) = -\sum_{x=1}^{N} (\log P[j(x,y)|i(x,y)] + \log P[i(x,y)|i(x-1,y), i'(x,y+1)]).$$

Necessary conditions for minimizing the free energy $F_{\text{seq}} =< U > -TH$ with respect to the row pmfs $\{P[\underline{I_y}|\mathbf{j}] \; \forall y\}$ subject to constraints ensuring $P[\underline{I_y}|\mathbf{j}]$ is a pmf, $\forall y$, are obtained by taking partial derivatives and setting the result to zero. This gives the

following coupled set of equations:

$$P[\underline{I_y} = \underline{i}|\mathbf{j}] = \frac{e^{-\frac{1}{T}(\sum_{\underline{i'}} P[\underline{I_{y+1}}=\underline{i'}|\mathbf{j}]U_y(\underline{i},\underline{i'})+\sum_{\underline{i''}} P[\underline{I_{y-1}}=\underline{i''}|\mathbf{j}]U_y(\underline{i''},\underline{i}))}}{Z(y)} \quad \forall \underline{i} \forall y. \tag{6.3}$$

Here, $Z(y)$ is the partition function, which ensures a pmf is obtained. As in basic MFA, the necessary optimality conditions (6.3) form the basis for fixed point iterations which, *at least in principle*, could be used to minimize Fseq at a given T. However, the FPIs based on (6.3) require computation and storage for $N|\mathcal{L}|^N$ values ! Moreover, ultimately, we are not really interested in $P[\underline{I_y}|\mathbf{j}]$ but rather in the *a posteriori* pmfs at each *site*, $P[I(x,y)|\mathbf{j}]$. While the FPIs based on (6.3) cannot be implemented *directly*, it was shown in [42] that they can be implemented *indirectly*, i.e. there is an *implementable* re-estimation procedure which, at each iteration, produces quantities that *determine* the row pmfs that *would have been obtained* had we re-estimated the row pmfs directly. In the segmentation context, these quantities (conveniently, from the standpoint of MAP segmentation) were the *a posteriori* site pmfs $\{P[I(x,y)|\mathbf{j}]\}$. These site pmfs were re-estimated using the Forward/Backward algorithm, more typically associated with inference in hidden Markov models. By re-estimating the site pmfs, the method *implicitly* implements the (non-implementable) re-estimation for the row pmfs. Significantly, the resulting method descends in the free energy F_{seq} [42].

Our JSC decoding approach follows fairly directly from the formulation and algorithm given in [42]. Similar to [42], we have a re-estimation procedure which *implicitly* implements the FPIs that are based on the necessary optimality conditions (6.3). This re-estimation approach is again based on the Forward/Backward algorithm. The principal

difference between the method in [42] and the JSC decoding method proposed here lies in the quantities that need to be re-estimated in order to determine/implicitly implement the FPIs over the row pmfs. It turns out that in the JSC decoding case the quantities that need to be re-estimated are both the (desired) *a posteriori site* pmfs $\{P[I(x,y)|\mathbf{j}]\}$, *as well as the pair-of-site pmfs* $\{P[I(x,y),I(x+1,y)|\mathbf{j}]\}$[4]. The Forward/Backward recursions are provided as follows. In particular, the *forward recursion* is given by

$$\begin{aligned}
\alpha^{(t)}_{(x,y)}[k] = \sum_{n=1}^{L} \alpha^{(t)}_{(x-1,y)}[n]\mathrm{EXP}(\frac{1}{T}(\log P[j(x,y)|I(x,y)=k] \\
+ \sum_{l=1}^{L} P^{(t)}[I(x,y+1)=l|\mathbf{j}] \log P[I(x,y)=k|I(x-1,y)=n, I(x,y+1)=l] \\
+ \sum_{l=1}^{L}\sum_{l'=1}^{L} P^{(t-1)}[I(x,y-1)=l; I(x-1,y-1)=l'|\mathbf{j}] \\
\cdot \log P[I(x,y-1)=l|I(x,y)=k; I(x-1,y-1)=l'])) \qquad (6.4)
\end{aligned}$$

while the backward recursion is given by

$$\begin{aligned}
\beta^{(t)}_{(x,y)}(k) = \sum_{n=1}^{L} \beta^{(t)}_{(x+1,y)}(n)\mathrm{EXP}(\frac{1}{T}(\log P[j(x+1,y)|I(x+1,y)=n] \\
+ \sum_{l=1}^{L} P^{(t)}[I(x+1,y+1)=l|\mathbf{j}] \\
\cdot \log P[I(x+1,y)=n|I(x+1,y+1)=l, I(x,y)=k] \\
+ \sum_{l=1}^{L}\sum_{l'=1}^{L} P^{(t-1)}[I(x+1,y-1)=l, I(x,y-1)=l'|\mathbf{j}] \\
\cdot \log P[I(x+1,y-1)=l|I(x+1,y)=n, I(x,y-1)=l'])) \qquad (6.5)
\end{aligned}$$

[4]The requirement that we re-estimate pmfs for pairs of sites is attributable to the second order Markov model for the encoded image.

The site probability then can be obtained by

$$p^{(t)}[I(x,y)=k|\mathbf{j}] = \sum_{\underline{i}\in\mathcal{L}^N : I(x,y)=k} P^{(t)}[\underline{I_y}=\underline{i}|\mathbf{j}] = \frac{\alpha^{(t)}_{(x,y)}(k)\beta^{(t)}_{(x,y)}(k)}{\sum_{l=1}^{L}\alpha^{(t)}_{(x,y)}(l)\beta^{(t)}_{(x,y)}(l)} \tag{6.6}$$

The pair-of-site probabilities can also be obtained as follows

$$P^{(t)}[I(x,y)=k, I(x+1,y)=n|\mathbf{j}] = \frac{P^{(t)}[I(x,y)=k, I(x+1,y)=n,\mathbf{j}]}{\sum_{k=1}^{L}\sum_{n=1}^{L} P^{(t)}[I(x,y)=k, I(x+1,y)=n,\mathbf{j}]} \tag{6.7}$$

where

$$\begin{aligned} P^{(t)}[I(x,y)=k, I(x+1,y)=n,\mathbf{j}] &= \alpha^{(t)}_{(x,y)}[k]\mathrm{EXP}\big(\frac{1}{T}(\log P[j(x+1,y)|I(x+1,y)=n] \\ &+ \sum_{l=1}^{L} P^{(t)}[I(x+1,y+1)=l|\mathbf{j}] \\ &\cdot \log P[I(x+1,y)=n|I(x+1,y+1)=l, I(x,y)=k] \\ &+ \sum_{l=1}^{L}\sum_{l'=1}^{L} P^{(t-1)}[I(x+1,y-1)=l, I(x,y-1)=l'|\mathbf{j}] \\ &\cdot \log P[I(x+1,y-1)=l|I(x+1,y)=n, I(x,y-1)=l'])\big)\beta^{(t)}_{(x+1,y)}[n] \end{aligned} \tag{6.8}$$

The computed Forward and Backward quantities determine at each iteration both the site *and* the pair-of-site *a posteriori* pmfs[5]. When implemented in a serial fashion (i.e. with one row updated at a time, with the remaining rows held fixed), one can show, similar

[5]For a more detailed account of this generalized MFA approach, see [42].

to the developments in [42], that the resulting method descends in the free energy F_{seq}. The resulting *a posteriori* probabilities at each site are the quantities of interest, needed to implement maximum *a posteriori* (MAP) detection or MMSE estimation.

6.6 Results and Conclusion

Our experiments have been performed on synthetic Markov random field images. The images were scalar quantized in the spatial domain to 2 bits per pixel. We compared a variety of techniques, including: 1) naive decoding, where $\hat{i}(x,y) = j(x,y)$; 2) the non-iterative MMSE decoder from [40], with sequence decoding applied separately for each row. Accordingly, the simpler model $P[i(x,y)|i(x-1,y)]$ must be used here; 3) a greedy, non-iterative decoder that applies sequence MAP decoding to each row, given the decoding result from the previous row; 4) the hillclimbing method from [7]. This approach is similar to the proposed method but iterates *dynamic programming* sweeps, rather than F/B sweeps. This method ascends in L. Thus, when initialized by method 3), this method is guaranteed to perform at least as well as method 3). However, method 4) is only suitable for MAP detection (MMSE estimation typically achieves better results [40].); 5) MMSE estimation, based on serial and parallel versions of basic MFA; 6) MMSE estimation, based on serial and parallel versions of the new MFA method, developed from [42]. All the mean-field optimizations (methods 5) and 6)) were performed at fixed $T = 1$, i.e. without annealing, in order to reduce complexity[6].

[6]Use of annealing has been observed to improve the performance somewhat; however, the computational complexity required to achieve these gains, relative to optimization at fixed $T = 1$, may be as much as 10-30 times higher.

The methods have been enumerated in order of increasing complexity – e.g., per iteration we found that the new MFA methods are ~ 4 times more complex per iteration than basic MFA, which in turn is ~ 4 times more complex than hillclimbing decoding[7]. As indicated in Fig. 1, the increased complexity "buys" gains in decoding accuracy, especially at the higher BERs. Basic MFA is seen, at BER=0.1, to gain more than 1 dB in PSNR over MMSE and hillclimbing-based decoding. The new sequence-based MFA methods gain $\sim$ an additional 0.4 dB. The complexity/decoding accuracy tradeoff can be gauged to identify the decoding method of choice for a given application.

[7] All methods were implemented, without code optimization, in C on a Sparc10.

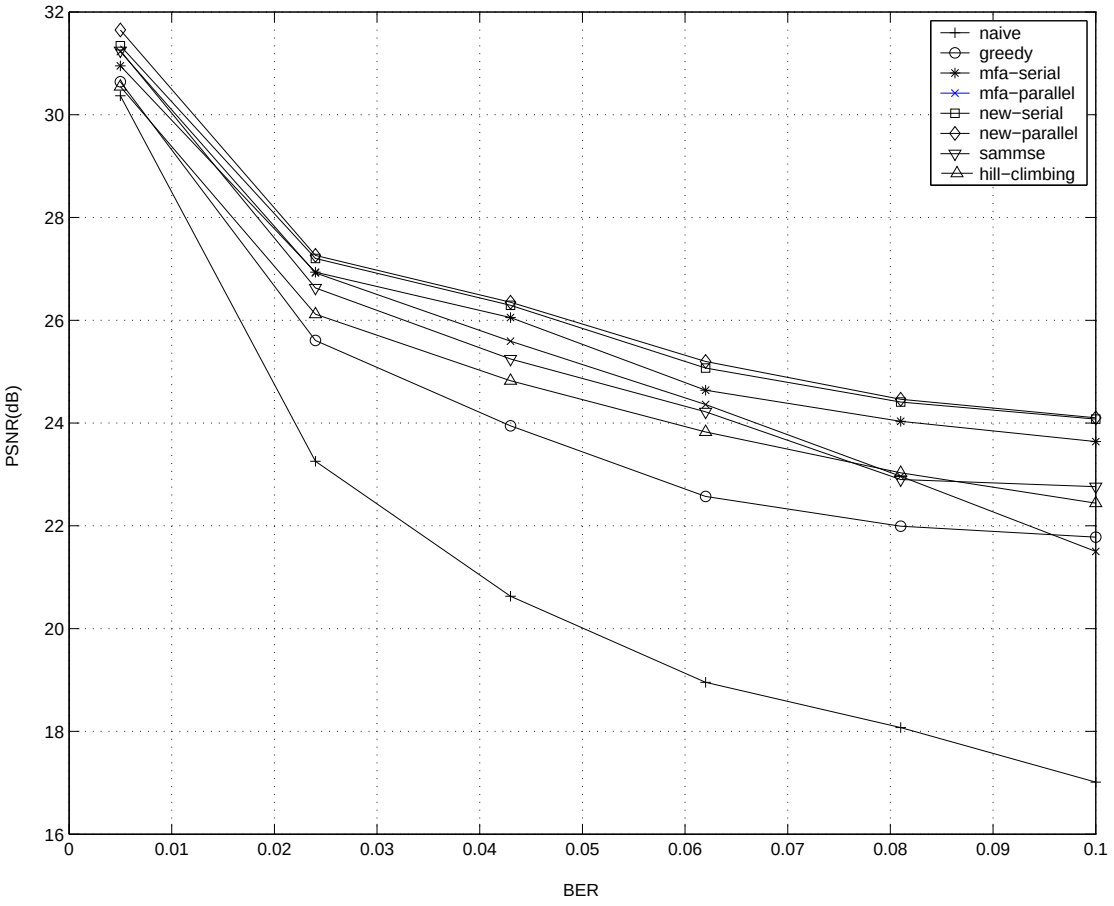

Fig. 6.1. JSC image decoding results over a binary symmetric channel

Chapter 7

Conclusion

Due to the exponential increase in computational complexity when increasing the order of a hidden Markov model in JSC decoding, one typically propose using a JSC decoder with order less than 2 or 3. However, higher order conditioning equals better performance in a JSC decoder due to the residual redundancy resulted by using a suboptimal source encoder. In this thesis, we proposed several novel techniques to bridge the gap of performance-complexity tradeoff in a JSC decoder (SAMMSE) based on Markov source modeling. One of these methods was further applied to achieve high order conditional entropy constrained encoding.

In chapter 2, we showed higher order conditioning can be achieved using the IIS algorithm. We also showed using higher order conditioning on the noisy channel symbols does not incur a large complexity increase in JSC decoding. In chapter 4, we further extended this idea to high order conditional entropy constrained encoding, in order to reduce the large training data set required to ensure good learning of the source model. In chapter 3, we proposed another bridging technique which is based on a least squares design principle. This approach uses a training data set to train the FIR least square filter taps to minimize the mean square error at the output of a low order JSC decoder. This effectively increases the memory of a JSC decoder without a high computational

complexity and achieves good decoding performance. In chapter 5, we also tried modeling the source symbol sequence as a hidden Markov model as opposed to the Markov model usually used. We were able to gain decoding performance compared to a standard JSC decoder (SAMMSE) with similar memory requirement, although our computational complexity is higher. A standard JSC decoder (SAMMSE) with similar computational complexity would produce better decoder result but not practical to implement due to its large memory requirement. We saw a consistent performance gain in our experiments when testing within training set. However, this performance gain is not consistent when testing outside the training set. Other possible extensions/applications need to be investigated on this method. In chapter 6, an extension of standard mean field annealing is applied to the Markov mesh modeling of image source in JSC decoding. We showed an iterative algorithm with relaxed independence assumption (independence is only between rows of image) to capture more residual redundancy used in combating noisy channels.

Appendix

First Hidden Markov Source Model

In this appendix, we will show a detailed formulation for the first hidden Markov source model introduced in Chapter 5. The state diagram of this model is shown in figure 5.2.

A.1 Learning

In this section, we will show how the two model parameters ($P[S_t = m|S_{t-1} = r]$, $P[I_t = k|S_t = l, I_{t-1} = m]$) are being learned via our iterative update equations. We first initialize the parameters of hidden Markov source model randomly due to the fact that the hidden states that generates the source symbol sequence have no underlying meaning that would suggest another initialization. We then need to calculate both the forward variable $\underline{\alpha}$ and backward variable $\underline{\beta}$ using recursive equations (A.1) and (A.2), respectively. After the forward variable $\underline{\alpha}$ and backward variable $\underline{\beta}$ are calculated, they are to be used in the update equations (A.3)(A.6) in learning the model. The same set of $\underline{\alpha}$ and $\underline{\beta}$ variables are then used to measure the likelihood(A.12) improvement for whether to stop the learning process. We also used these $\underline{\alpha}$ and $\underline{\beta}$ variables to measure the entropy(A.10) of our proposed model as an indication of how well a lossless compression would perform if an entropy coding (VLC) was applied. Our algorithm was extended from the idea of Expectation Maximization techniques developed by Baum et al.[3].

A.1.1 Update Equations

In this section, we will first show the formulation of the forward and backward variables. Then, we will show how these variables are used in our model parameter update equations.

The forward variable $\underline{\alpha}$ is calculated recursively from $t = 1$ to $t = T - 1$, given by

$$\begin{aligned} &\alpha_t(S_t = m) = P[i_0, \ldots, i_t, S_t = m] \\ &= \big[\sum_r \alpha_{t-1}(S_{t-1} = r) P[S_t = m | S_{t-1} = r]\big] P[i_t | S_t = m, i_{t-1}]. \end{aligned} \tag{A.1}$$

$\underline{\alpha}$ is the probability of the causal part of source symbol sequence joint with the possible current hidden state. α_t is calculated in a special way when $t = 0$ in the same fashion as in the standard SAMMSE decoder case[40].

The backward variable $\underline{\beta}$ is calculated recursively from $t = T - 2$ to $t = 0$, given by

$$\begin{aligned} &\beta_t(S_t = m) = P[i_{t+1}, \ldots, i_{T-1} | S_t = m] \\ &= \sum_r P[S_{t+1} = r | S_t = m] P[i_{t+1} | S_{t+1} = r, i_t] \beta_{t+1}(S_{t+1} = r). \end{aligned} \tag{A.2}$$

It is set to one when $t = T - 1$. $\underline{\beta}$ is the probability of the anti-causal part of the source symbol sequence given the possible current hidden state.

We take the $\underline{\alpha}$ and $\underline{\beta}$ from (A.1) and (A.2) to be used in our learning equations (A.3) and (A.6). The set of learning equations are to iteratively update the parameters of the model with the increasing likelihood. After each iteration of the learning equation, a

new $\underline{\alpha}$ variable and $\underline{\beta}$ variable from (A.1) and (A.2) needs to be recalculated using these updated model parameters from (A.3) and (A.6) for the next iteration. The likelihood (A.12) also needs to be recalculated after each iteration for the stopping criterion. The iteration is stopped when the difference of likelihood between two consecutive iterations are smaller than 0.5 and more than 50 iterations have been done.

The **first model parameter** is re-estimated by

$$P^{(n+1)}[S_t = m|S_{t-1} = r] = \frac{\sum_{t=0}^{T-1} P^{(n)}[S_t = m, S_{t-1} = r|\underline{i}]}{\sum_{t=0}^{T-1} P^{(n)}[S_{t-1} = r|\underline{i}]} \tag{A.3}$$

where superscript (n) denotes the counter/index for iterations. Its numerator and denominator can be simplified to a function of the current $\underline{\alpha}$ and $\underline{\beta}$ from (A.1) and (A.2) and the model parameters from the previous iteration. The probability in the numerator is given by,

$$P^{(n)}[S_t = m, S_{t-1} = r|\underline{i}] =$$

$$\frac{\alpha_{t-1}(S_{t-1} = r)\beta_t(S_t = m)P^{(n)}(S_t = m|S_{t-1} = r)P^{(n)}[i_t|i_{t-1}, S_t = m]}{\sum_{m',r'} \alpha_{t-1}(S_{t-1} = r')\beta_t(S_t = m')P^{(n)}(S_t = m'|S_{t-1} = r')P^{(n)}[i_t|i_{t-1}, S_t = m']}. \tag{A.4}$$

The probability in the denominator is given by,

$$P^{(n)}[S_{t-1} = r|\underline{i}] = \frac{\alpha_{t-1}(S_{t-1} = r)\beta_{t-1}(S_{t-1} = r)}{\sum_{r'} \alpha_{t-1}(S_{t-1} = r')\beta_{t-1}(S_{t-1} = r')}. \tag{A.5}$$

The **second model parameter** is re-estimated by

$$P^{(n+1)}[I_t = k|S_t = l, I_{t-1} = m] = \frac{\sum_{t=0}^{T-1} P^{(n)}[I_t = k, S_t = l, I_{t-1} = m|\underline{i}]}{\sum_{t=0}^{T-1} P^{(n)}[S_t = l, I_{t-1} = m|\underline{i}]}. \quad \text{(A.6)}$$

Its numerator and denominator can also be simplified to a function of current $\underline{\alpha}$ and $\underline{\beta}$ from (A.1) and (A.2) and the model parameters from the previous iteration. The probability in the numerator is given by,

$$P^{(n)}[I_t = k, S_t = l, I_{t-1} = m|\underline{i}] = \frac{\alpha_t(S_t = l)\beta_t(S_t = l)\delta(i_t - k)\delta(i_{t-1} - m)}{\sum_{l'}\alpha_t(S_t = l')\beta_t(S_t = l')}. \quad \text{(A.7)}$$

The probability in the denominator is given by,

$$P^{(n)}[S_t = l, I_{t-1} = m|\underline{i}] = \frac{\alpha_t(S_t = l)\beta_t(S_t = l)\delta(i_{t-1} - m)}{\sum_{l'}\alpha_t(S_t = l')\beta_t(S_t = l')}. \quad \text{(A.8)}$$

The **initial model parameter** $P[I_t = k|S_t = l]$ is updated in the following equation:

$$P^{(n+1)}[I_t = k|S_t = l] = \frac{\sum_{t=0}^{T-1}\sum_{m} P^{(n)}[I_t = k, S_t = l, I_{t-1} = m|\underline{i}]}{\sum_{t=0}^{T-1}\sum_{m} P^{(n)}[S_t = l, I_{t-1} = m|\underline{i}]}. \quad \text{(A.9)}$$

A.2 Entropy and Likelihood

There are two measurements for our hidden Markov source model. First is the entropy measure. Second is the likelihood measure. The source entropy is the minimum rate of lossless compression. It is therefore a good measurement of how much the source could be compressed losslessly if an actual VLC code was used. In this section, we will show that we are able to achieve infinite memory in a constrained form in contrast to a standard approach. The likelihood measure is used to determine the stopping point of our learning iterations.

A.2.1 Entropy

The entropy measure of our hidden Markov source model is given by

$$H\big(P[I_t = l|i_{t-1}, \ldots, i_0]\big) = \frac{\sum_{t=0}^{T-1} \sum_{l} P[I_t = l|i_{t-1}, \ldots, i_0] \log P[I_t = l|i_{t-1}, \ldots, i_0]}{T}, \tag{A.10}$$

where

$$P[I_t = l|i_{t-1}, \ldots, i_0] = \frac{\sum_{m,r} \alpha_{t-1}(S_{t-1} = r) P[S_t = m|S_{t-1} = r] P[I_t = l|S_t = m, i_{t-1}]}{\sum_{r} \alpha_{t-1}(S_{t-1} = r)}. \tag{A.11}$$

Equation (A.11) shows the advantage of this hidden Markov source model achieving infinite memory in a constrained form compared to the finite memory $P[I_t|I_{t-1}, I_{t-2}]$ used in the second order Markov source model.

A.2.2 Likelihood

The likelihood is measured after each iteration of learning equations to validate our formulation and provide a stopping criterion for our learning. This likelihood function is given by

$$\begin{aligned} P[\underline{i}|\lambda] &= \sum_{m} \tilde{\alpha}_t(S_t = m)\tilde{\beta}_t(S_t = m) \\ &= \sum_{m} \tilde{\alpha}_{T-1}(S_{T-1} = m), \end{aligned} \tag{A.12}$$

where $\tilde{\alpha}_t$ and $\tilde{\beta}_t$ represent the values before renormalization. The $\tilde{\alpha}_t$ and $\tilde{\beta}_t$ have to be renormalized by

$$\begin{aligned} \alpha_t(S_t = m) &= \frac{\hat{\alpha}_t(S_t = m)}{\sum_{m'} \hat{\alpha}_t(S_t = m')} \\ \beta_t(S_t = m) &= \frac{\hat{\beta}_t(S_t = m)}{\sum_{m'} \hat{\beta}_t(S_t = m')} \end{aligned} \tag{A.13}$$

at every symbol time in equation (A.1) and (A.2) to avoid numerical underflow, where $\hat{\alpha}_t$ denotes the recursive version of renormalized $\tilde{\alpha_t}$ at symbol time t. In order to find the log likelihood of $P[\underline{i}|\lambda]$, we need to keep the scale factors used in α_t at every symbol time in order to unscale the α_t at $t = T-1$. We have not seen anyone else being able to calculate the likelihood for source symbol sequence using the Forward/Backward algorithm in any publications. From the property of $\underline{\alpha}$ variable recursion, we realized that in order to unscale α_{T-1} we need to multiply all the scale factors $scale(t) = \sum_{m'} \hat{\alpha}_t(S_t = m')$ from

$t = 0$ to $t = T - 1$. The likelihood then becomes

$$P[\underline{i}|\lambda] = \prod_{t=0}^{T-1} scale(t) \sum_{m} \alpha_{T-1}(S_{T-1} = m). \tag{A.14}$$

Since $\sum_m \alpha_{T-1}(S_{T-1} = m) = 1$ after scaling, the log likelihood is given by

$$\log(P[\underline{i}|\lambda]) = \sum_{t=0}^{T-1} \log(scale(t)) \tag{A.15}$$

where the scale factors of $\underline{\alpha}$ are obtained from (A.1).

A.3 Inference

In this section, a JSC decoding algorithm is designed for our first hidden Markov source model to output the posteriori probabilities of transmitted symbol sequence by using the noisy symbol sequence and the converged updated model parameters. A conditional mean estimator[40] is then used to estimate the transmitted quantization levels.

A.3.1 JSC decoder

A *different* set of forward and backward variables (not the same as the forward and backward variables in learning) is used here to determine the *a posteriori*i probabilities. In order to make this distinction clear, we use $\underline{\breve{\alpha}}$ and $\underline{\breve{\beta}}$ to denote the forward and backward variables in the inference section.

The *a posteriori*i probabilities are obtained by

$$\begin{aligned} P[I_t = q|\underline{j}] &= \sum_k P[I_t = q, S_t = k|\underline{j}] \\ &= \sum_k \frac{\check{\alpha}_t(I_t = q, S_t = k)\check{\beta}_t(I_t = q, S_t = k)}{\sum_{q'}\sum_{k'}\check{\alpha}_t(I_t = q', S_t = k')\check{\beta}_t(I_t = q', S_t = k')} \quad \forall t, \end{aligned} \tag{A.16}$$

where $\check{\alpha}_t(I_t = q, S_t = k)\check{\beta}_t(I_t = q, S_t = k) = P[I_t = q, S_t = k, \underline{j}]$.

The $\check{\alpha}_t$ variable is calculated recursively by

$$\begin{aligned} &\check{\alpha}_t(I_t = q, S_t = k) = P[j_0, \ldots, j_t, I_t = q, S_t = k] \\ &= \big[\sum_m \sum_l \check{\alpha}_{t-1}(I_{t-1} = l, S_{t-1} = m)P[S_t = k|S_{t-1} = m]\cdot \\ &P[I_t = q|S_t = k, I_{t-1} = l]\big]P[j_t|I_t = q] \end{aligned} \tag{A.17}$$

from $t = 1$ to $t = T-1$. $\check{\alpha}$ is the probability of the causal part of received symbol sequence joint with the possible current hidden state and decoded symbol. $\check{\alpha}_t$ is calculated in a special way when $t = 0$ in the same fashion as in a standard SAMMSE decoder[40].

The $\check{\beta}_t$ variable is calculated recursively by

$$\begin{aligned} &\check{\beta}_t(I_t = q, S_t = k) = P[j_{t+1}, \ldots, j_{T-1}|I_t = q, S_t = k] \\ &= \sum_m \sum_l P[I_{t+1} = l|S_{t+1} = m, I_t = q]P[S_{t+1} = m|S_t = k]P[j_{t+1}|I_{t+1} = l]\cdot \\ &\check{\beta}_{t+1}(I_{t+1} = l, S_{t+1} = m) \end{aligned} \tag{A.18}$$

from $t = T - 2$ to $t = 0$. It is set to one when $t = T - 1$. $\check{\beta}$ is the probability of the anti-causal part of the received symbol sequence given the possible current hidden state and decoded symbol.

Bibliography

[1] L.M. Arslan and J.H.L. Hansen. Selective Training for Hidden Markov Models with Applications to Speech Classification. *IEEE Transactions on Speech and Audio Processing*, 7(1):46–54, Jan. 1999.

[2] R. Bauer and J. Hagenauer. Turbo-FEC/VLC-Decoding and its Application to Text Compression. *Conference on Information Sciences and Systems*, 1:WA6–6–WA6–11, March 2000.

[3] L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *Annals of Math. and Stat.*, 41:164–171, 1970.

[4] A. Berger, S. Della Pietra, and V. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, pages 22:39–68, 1996.

[5] D. Van Den Bout and T.K. Miller. Graph Partitioning using Annealed Neural Networks. *IEEE Transactions on Neural Networks*, pages 192–203, 1990.

[6] P. Bunyaratavej. *Iterative Hillclimbing Optimization Techniques for Transform Image Encoding/Decoding and for Image Segmentations*. PhD thesis, The Pennsylvania State University, May 2002.

[7] P. Bunyaratavej and D.J. Miller. An Iterative Hillclimbing Algorithm for Discrete Optimization on Images: Application to Joint Encoding of Image Transform Coefficients. *IEEE Signal Processing Letters*, 9(2):46–50, Feb. 2002.

[8] P. Cheeseman. A Method of Computing Generalized Baysian Probability Values for Expert Systems. *Proceedings of the Eighth International Joint Conference on AI*, 1:198–202, 1983.

[9] K.-H. Chei and K.-P. Ho. Design of Optimal Soft Decoding for Combined Trellis Coded Quantization/Modulation in Rayleigh Fading Channel. *IEEE Intl. Conf. on Acoustics, Speech, and Sig. Proc.*, pages 2633–2636, 2000.

[10] P.A. Chou and T. Lookabaugh. Conditional Entropy-Constrained Vector Quantization of Linear Predictive Coefficients. *ICASSP*, 1:197–200, April 1990.

[11] K. M. Chugg, C. Xiaopeng, A. Ortega, and C. Cheng-Wei. An Iterative Algorithm for Two-Dimensional Digital Least Metric Problems with Applications to Digital Image Compression. *ICIP*, 2:722–726, Oct. 1998.

[12] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* A Wiley-Interscience publication, 1991.

[13] R.E. Van Dyke and D.J. Miller. Transport of Wireless Video using Separate Concatenated and Joint Source-channel Coding. *IEEE Proceedings*, 87(10):1734–1750, Oct. 1999.

[14] S. Emami and S. Miller. DPCM Picture Transmission over Noisy Channel with the Aid of a Markov Model. *IEEE Transactions on Image Processing*, 4(11):1473–1479, Nov. 1995.

[15] T. Fazel and T. Fuja. Robust Transmission of MELP-compressed Speech: an Illustrative Example of Joint Source-channel Decoding. *IEEE Transactions on Communications*, 51(6):973–982, June 2003.

[16] T.R. Fischer and M. Wang. Entropy-Constrained Trellis-Coded Quantization. *IEEE Transactions on Information Theory*, 38(2):415–426, March 1992.

[17] S. Forchhammer and T.S. Rasmussen. Adaptive Partially Hidden Markov Models with Applications to Bilevel Image Coding. *IEEE transactions on Image Processing*, 8(11):1516–1526, Nov. 1999.

[18] G.D. Forney. The Viterbi Algorithm. *IEEE Proceedings*, 61:268–278, March 1973.

[19] J. Garcia-Frias and J. D. Villasenor. Combining Hidden Markov Source Models and Parallel Concatenated Codes. *IEEE Communication Letters*, 1(4):111–113, July 1997.

[20] J. Garcia-Frias and J.D. Villasenor. Joint Turbo Decoding and Estimation of Hidden Markov Sources. *IEEE Journal on Selected Areas in Communications*, 19(9):1671–1679, Sept. 2001.

[21] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic, 1992.

[22] W.B. Gevarter. Automatic Probabilistic Knowledge Acquisition from Data. *Technical Report NASA technical memorandum 88224, NASA*, 1986.

[23] A. Guyader, E. Fabre, C. Guillemot, and M. Rebert. Joint Source-Channel Turbo Decoding of Entropy-Coded Sources. *IEEE Journal on Selected Areas in Communications*, 19(9):1680–1696, Sept. 2001.

[24] M. H. Hayes. *Statistical Digital Signal Processing and Modeling.* Wiley, 1996.

[25] E. Herskovits and G. Cooper. Kutato. An Entropy-Driven System for Construction of Probabilistic Expert Systems from Databases. *Uncertainty in AI*, 6:117–125, 1991.

[26] T. Hofmann, J. Puzicha, and J.M. Buhmann. Unsupervised Texture Segmentation in a Deterministic Annealing Framework. *IEEE Transactions on Patt. Anal. and Mach. Intell.*, 20:803–818, 1998.

[27] J.R. Deller Jr., J.G. Proakis, and J.H.L. Hansen. *Discrete-time Processing of Speech Signals.* Macmillan Publishing, New York, 1993.

[28] F. Kossentini, W.C. Chung, and M.J.T. Smith. Conditional Entropy-Constrained Residual VQ with Application to Image Coding. *IEEE Transactions on Image Processing*, 5(2):311–320, Feb. 1996.

[29] I. Kozintsev, R. Koetter, and K. Ramchandran. A Factor Graph Framework for Joint Source-Channel Decoding of Images. *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers, 1999*, 1(24–27):321–325, Oct. 1999.

[30] S.J. Lee and C.W. Lee. Conditional Entropy-Constrained Trellis-Searched Vector Quantization for Image Compression. *Signal Processing: Image Communication*, 8:99–104, 1996.

[31] R. Link and S. Kallel. Markov Model Aided Decoding for Image Transmission using Soft-Decision-Feedback. *IEEE Transactions on Image Processing*, 9(2):190–196, Feb. 2000.

[32] R. Link and S. Kallel. Optimal Use of Markov Model for DPCM Picture Transmission over Noisy Channels. *IEEE Transactions on Communications*, 48(10):1702–1711, Oct. 2000.

[33] S. P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28:127–135, March 1982.

[34] M. Malfait and D. Roose. Wavelet-based Image Denoising Using a Markov Random Field A Priori Model. *IEEE Transactions on Image Processing*, 6(4):549–565, April 1997.

[35] M. W. Marcellin. On Entropy-Constrained Trellis Coded Quantization. *IEEE Transactions on Communications*, 42(1):14–16, Jan. 1994.

[36] M.W. Marcellin and T.R. Fischer. Trellis Coded Quantization of Memoryless and Gauss-Markov Sources. *IEEE Transactions on Communications*, 38(1):82–93, Jan. 1990.

[37] B. Martins and S. Forchhammer. Lossless, Near-Lossless, and Refinement Coding of Bilevel Images. *IEEE Transactions on Image Processing*, 8(5):601–613, May 1999.

[38] R. Meir. On Deriving Deterministic Learning Rules from Stochastic System. *International Journal of Neural Systems*, 2:283–289, 1992.

[39] D.J. Miller, E. S.G. Carotti, Y. Wang, and J. C. de Martin. Joint Source-Channel Decoding of Predictively and Non-Predictively Encoded Sources: a Two-Stage Estimation Approach. *IEEE Transactions on Communications*. Submitted to.

[40] D.J. Miller and M.S. Park. 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, February 1998.

[41] D.J. Miller and L. Yan. Approximate Maximum Entropy Joint Feature Inference Consistent with Arbitrary Lower-End Probability Constraints: Application to Statistical Classification. *Neural Computation*, 12:2175–2207, 2000.

[42] D.J. Miller and Q. Zhao. A Sequence Based Extension of Mean-Field Annealing using the Forward/Backward Algorithm: Application to Image Segmentation. *IEEE Transactions on Signal Processing*, 51(10):2692–2705, Oct. 2003.

[43] A.H. Murad and T.E. Fuja. Joint Source-Channel Decoding of Variable-Length Encoded Sources. *Information Theory Workshop*, (22–26):94–95, June 1998.

[44] M. Park. *Joint Source-Channel Decoding Using Residual Redundancy Based on a Standard and Novel Estimation Techniques for Hidden Markov Models*. PhD thesis, The Pennsylvania State University, March 1999.

[45] M.S. Park and D.J. Miller. Low-Delay Optimal MAP State Estimation in HMM's with Application to Symbol Decoding. *IEEE Signal Processing Letters*, 4:289–292, October 1997.

[46] M.S. Park and D.J. Miller. Improved Image Decoding over Noisy Channels using Minimum Mean-Squared Estimation and a Markov Mesh. *IEEE Transactions on Image Processing*, 8(6):863–867, June 1999.

[47] M.S. Park and D.J. Miller. Joint Source-Channel Decoding for Variable-Length Encoded Data by Exact and Approximate MAP Sequence Estimation. *IEEE Transactions on Communications*, 48:1–6, Jan. 2000.

[48] N. Phamdo and N. Farvardin. Optimal Detection of Discrete Markov Sources over Discrete Memoryless Channels-Application to Combined Source-Channel Coding. *IEEE Transactions on Inform. Theory*, 40:186–193, 1994.

[49] L. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[50] L.R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of IEEE*, 77(2):257–286, Feb. 1989.

[51] A.V. Rao and K. Rose. Deterministically Annealing Design of Hidden Markov Model Speech Recognizers. *IEEE Transactions on Speech and Audio Processing*, 9(2):111–126, Feb. 2001.

[52] J. Rissanen. Universal Coding, Information, Prediction, and Estimation. *IEEE Transactions on Information Theory*, IT-30:629–636, July 1984.

[53] K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann, 2 edition, 2000.

[54] K. Sayood and J.C. Borkenhagen. Use of Residual Redundancy in the Design of Joint Source/Channel Coders. *IEEE Transactions on Communications*, 39(6):838–846, June 1991.

[55] C.E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423,623–656, 1948.

[56] J. Stuller and B. Kurz. Two-Dimensional Markov Representation of Sampled Images. *IEEE Transactions on Communications*, 24(10):1148–1152, Oct. 1976.

[57] K.P. Subbalakshmi and J. Vaisey. On the Joint Source-Channel Decoding of Variable-Length Encoded Sources:the BSC Case. *IEEE Transactions on Communications*, 49(12):2052–2055, Dec. 2001.

[58] W. Turin. MAP Decoding in Channels with Memory. *IEEE Transactions on Communications*, 48(5):757–763, May 2000.

[59] G. Ungerboeck. Trellis-Coded Modulation with Redundant Signal Sets Part I & II. *IEEE Communications Magazine*, 25(2):5–21, Feb. 1987.

[60] X. Wu, J. Wen, and W. H. Wong. Conditional Entropy Coding of VQ Indexes for Image Compression. *IEEE Transactions on Image Processing*, 8(8):1005–1013, Aug. 1999.

[61] W. Xu, J. Hagenauer, and J. Hollman. Joint Source-Channel Decoding Using the Residual Redundancy in Compressed Images. *Proc. ICC/SUPERCOMM'96*, pages 142–148, June 1996.

Vita

Yu-wei Wang was born in Taipei, Taiwan in 1974. In 1997, he received his B.S. degree in Electrical Engineering from the Tennessee Technological University. He then came to the Pennsylvania State University to continue his graduate study. In the fall of 1999, he received his M.S. degree in Electrical Engineering from the Pennsylvania State University, where he joined the Ph.D. program in the spring of 1999. He has been working in Prof. Miller's group since fall of 1999. He will be working as a color scientist in digital camera development at the Hewlett Packard company in January, 2004. His research interests are in the area of digital signal processing in communication applications, digital speech/image/video signal processing, joint source channel coding/decoding, data compression, and digital communications. He is a member of Kappa Mu Epsilon (Mathematics Honor Society) and Eta Kappa Nu (Electrical Engineering Honor Society).