The Pennsylvania State University

The Graduate School

Department of Electrical Engineering

ITERATIVE HILLCLIMBING OPTIMIZATION TECHNIQUES FOR TRANSFORM IMAGE ENCODING / DECODING AND FOR IMAGE SEGMENTATION

A Thesis in

Electrical Engineering

by

Piya Bunyaratavej

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2002

We approve the thesis of Piya Bunyaratavej.

Date of Signature

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	
George Kesidis Associate Professor of Electical Engineering	
Jia Li Assistant Professor of Statistics	
W. Kenneth Jenkins Professor of Electrical Engineering Head of the Department of Electrical Engineering	

Abstract

An iterative hillclimbing optimization technique was introduced in this thesis. It was used to tackle many index assignment problems, i.e. transform image encoding, noisy image decoding, and image segmentation. We first studied basic transform image coding techniques, then introduced an iterative algorithm which has a hillclimbing property on the cost function. We then extended the algorithm to hyperspectral image coding. We realized that the algorithm can be generalized to other applications as well. We applied the iterative hillclimbing idea to noisy channel image decoding. We also investigated a Turbo-like joint source-channel decoding technique, which is another kind of iterative decoding. Lastly, we re-investigated the image segmentation application, using the iterative hillclimbing algorithm. The hillclimbing method inspired development of another iterative algorithm which is an extension to mean-field annealing, with applications both to image decoding and image segmentation. The hillclimbing algorithm at the heart of this thesis thus yielded several promising offshoot directions for continuing research.

Table of Contents

List of Figures

Acknowledgments

I am most grateful to my thesis advisor, David J. Miller, for his guidance, patience, encouragement and support he has shown me during my time working with him. His insight and enthusiasm in research subjects have a profound impact to how I work. This will definitely carry with me into the future. I am also grateful and indebted to all committee members: Dr. Doherty, Dr. Metzner, Dr. Kesidis, and Dr. Li, for serving on my committee and insightful commentary.

I would like to dedicate this thesis to my parents in Thailand who always give me a support I need, even though they live far away.

Chapter 1

Introduction

1.1 Problem Statement

The image compression problem can be viewed as an index assignment problem, where an encoder searches for an optimum index to represent a source sample. For example, consider an entropy-constrained quantization. An encoder (quantizer) searches for quantization indices to represent image samples. This can be thought of as a discrete optimization problem, where optimum indices are searched from a set of possible solutions. There is a high spatial redundancy within an image. Usually, a high-order probability model is used to capture image redundancies. An optimum image encoder, using exhaustive search algorithm, has an intractable computational complexity. Thus, an alternative is to use suboptimal encoders. A greedy encoder is one of many suboptimal encoders that has a lower computational complexity. We studied and proposed an alternative hillclimbing-based encoding scheme that has a tradeoff between complexity and performance.

Noisy image decoding can be thought of as an index assignment problem as well, where a decoder assigns an index to a received (noisy) sample. Similar to image encoding problem, an optimal decoder (based on 2-dimensional probability models for the encoded image) has very high computational complexity. With a slight modification

to the proposed hillclimbing algorithm for image encoding, we proposed an alternative image decoding technique.

Another index assignment problem is image segmentation. A popular approach to image segmentation is a mean-field technique [11],[25],[31], which is very computationally demanding. Applying the iterative approach proposed in this thesis shows minimal improvement. An alternative approach was developed collaboratively [50],[52], and investigated in a preliminary fashion in this thesis. This approach is a natural extension of the proposed hillclimbing algorithm.

1.2 Technical Background

This section provides basic backgrounds which are fundamental to this thesis.

1.2.1 Basic Optimization Techniques

A goal of any optimization problem is to find the best solution out of a space of possible solutions. For discrete optimization problems, solutions are searched from a set of discrete values. There are three main searching techniques: exhaustive, greedy and stochastic search.

- Exhaustive search guarantees to find a globally optimal solution. Since the computational complexity is the highest, it is suitable only for a small search space.

- Greedy optimization divides the problem into smaller pieces and searches for a solution for a small piece one at a time. The obtained solution is optimal for

that particular small piece, but the overall solution is not guaranteed to be optimum, both globally and locally. It is suitable for a larger search space since the computational complexity is less than that of exhaustive search.

- Stochastic search finds a solution by using gradient descent search but uphill transitions are sometimes allowed. These uphill transitions help avoiding a (poor) locally optimal solution. The better the current solution, the less likely uphill transitions are allowed.

  - Simulated Annealing is shown to converge in probability to the global minimum, even for non-convex problems [26]. One drawback for simulated annealing is that the computational complexity is very high.

  - Mean-Field Annealing uses a deterministic approximation to simulated annealing, by attempting to average over the statistics of the annealing process, to speed up the convergence.

These three basic optimization techniques will be used in different parts of this thesis.

1.2.2 Trellis Coded Quantization

Trellis coded quantization (TCQ) uses Ungerboeck's set partitioning ideas from trellis coded modulation to achieve performance comparable to that of conventional trellis coding with a fraction of the complexity of conventional trellis coding [44]. TCQ is a type of finite-state quantizer that employs a scalar codebook and a dynamic programming algorithm in encoding. Trellis branches are labeled with subsets of the codebook.

Consider the TCQ design phase. Let $\underline{x} \equiv (x_1, x_2, \ldots, x_N)$, $x_n \in \mathcal{R}$ be the training data sequence. Let $\underline{i} \equiv (i_1, i_2, \ldots, i_N)$ be a set of encoded indices. Let $\underline{y} \equiv (y_1, y_2, \ldots, y_N)$ be a reconstructed sequence. Given the initial codebook, the encoder finds the path through the trellis to minimize $D = \sum_n d(x_n, y_n)$, where $d(\cdot, \cdot)$ is a specified scalar distortion measure, and where the quantization choices for y_n are constrained by the trellis structure. The codebook is then updated to minimize the overall distortion via a centroid rule applied to each quantization level. This design process is a trellis version of the Lloyd algorithm, attributed to Stewart and Gray [63].

Because TCQ uses an expanded signal set and encodes a long sequence of samples jointly, it is a much better quantizer than a corresponding scalar quantizer at the same rate. In fact, evaluation of the asymptotic vector quantization (VQ) bound indicates that no VQ of dimension less than 69 can exceed the MSE performance of 256 state TCQ, for a memoryless uniform source [44]. TCQ is also relatively insensitive to channel noise. TCQ maps an input symbol to a pair of integers (i, j) according to the current symbol and previous trellis state, where an integer i decides the subset and trellis state and an integer j selects a codeword within the specified subset. If the channel error corrupts j, it only leads to codewords within the same subset, and there is no effect on the trellis state, which means the error will not propagate to the subsequent output. If i is corrupted by the channel, the wrong subset is picked and the trellis state is incorrect. However, due to the trellis structure, this effect will not be propagated beyond $1 + log_2 N$ outputs, where N is the number of states. Fixed-rate TCQ needs at least one bit to specify the subset. Therefore, the coding rate cannot be less than 1 bit per sample.

Entropy-constrained trellis coded quantization (ECTCQ) combines the generalized Lloyd algorithm for trellis coded design with the entropy-constrained scalar quantizer design algorithm [29], subject to the TCQ structure [22]. In this case, the encoder tries to minimize the MSE but subject to a rate constraint by minimizing the Lagrangian cost function $J = \sum_n (d(x_n, y_n) + \lambda l(y_n))$, where $l(y_n)$ is the number of bits used by the variable-length code to represent y_n. The design steps are similar to fixed-rate TCQ.

Conditional Entropy-Constrained Trellis Coded Quantization (CECTCQ) tries to capture statistical dependencies either within a source (for 1-D sources) or between sources (for images) [36]. Consider a 1-D sequence of source samples. Conditional ECTCQ uses the conditional probabilities $P[i_n|i_{n-1}, S_{n-1}]$, where S_{n-1} is the TCQ state at time $n-1$. The encoder finds the best codeword by minimizing the Lagrangian cost function $J = \sum_n (d(x_n, y_n) - \lambda \log_2 P[i_n|i_{n-1}, S_{n-1}])$. CECTCQ can also benefit the coding of multiple sources that have dependencies between them, in particular, the multiple (transform coefficient) sources obtained from image data which will be discussed in chapter 2. In this case, the CECTCQ encoder can be chosen to minimize $J = \sum_n (d(x_{k,n}, y_{k,n}) - \lambda \log_2 P[i_{k,n}|i_{k,n-1}, i_{k-1,n}, S_{n-1}^k, S_n^{k-1}])$, where letter k represents the kth source. Note that the self-information term incorporates dependencies both *within* the kth source (between $n-1$ and n), and *between* the kth and $(k-1)$th sources. It is obvious that if there are no dependencies, conditional probabilities become marginal probabilities, and then CECTCQ becomes ECTCQ.

TCQ will be used in chapter 2 and 3 as a source encoder because it can be categorized as a state of the art or near state-of-the-art encoder when used in wavelet and transform coding frameworks.

1.2.3 Hidden Markov Model

A Markov model is a finite-state machine with stochastic transition in which the sequence of states is a Markov chain. Let I_n be a random state at time n, and its realization given by i_n. Also, let $\underline{I}$ be a random state sequence with its realization $\underline{i}$. The probability of any sequence $\underline{I}$ is

$$P[\underline{I} = \underline{i}] = P[I_1 = i_1] \prod_{n=2}^{n=N} P[I_n = i_n | I_{n-1} = i_{n-1}, I_{n-2} = i_{n-2}, \ldots, I_1 = i_1] \quad (1.1)$$

where N the total sequence length. For a first-order Markov model, (1.1) is reduced to

$$P[\underline{I} = \underline{i}] = P[I_1 = i_1] \prod_{n=2}^{n=N} P[I_n = i_n | I_{n-1} = i_{n-1}]. \quad (1.2)$$

A hidden Markov model (HMM) is referred to extensively in this thesis. While each output of a Markov model corresponds to an observed event, in an HMM, the Markovian state sequence is not observed – it is hidden, i.e. only the output sequence, $\underline{Z}$, can be observed. Given the observed sequence, one can determine 1) the most probable state sequence, $\hat{\underline{i}}$, or 2) the most probable state at a particular time, $\hat{i}_n$ (or estimate the probability distribution over these states).

To find the most probable state sequence, a popular solution is to use the Viterbi algorithm [23], which is based on dynamic programming methods. Let $\delta_n[l]$ be the probability metric along a single path, at time n, which accounts for the first n observations and ends in state l. And let $\psi_n[l]$ be the array that keeps the state sequence up to time

n and ends in state l. The following algorithm searches for $\hat{\underline{i}} = \arg\max_{\underline{i}} P[\underline{I} = \underline{i}|\underline{Z} = \underline{z}]$.

$$\begin{aligned}
\delta_1[l] &= P[I_1 = l]P[Z_1 = z_1|I_1 = l], \quad l = 1, \ldots, L \\
\delta_n[l] &= \max_{1 \le k \le L} \left[\delta_{n-1}[k]P[I_n = l|I_{n-1} = k]\right] P[Z_n = z_n|I_n = l], \\
&\quad l = 1, \ldots, L, \quad n = 2, \ldots, N \\
\psi_n[l] &= \arg \max_{1 \le k \le L} \left[\delta_{n-1}[k]\mathrm{P[I_n = l|I_{n-1} = k]}\right], l = 1, \ldots, L, \quad n = 2, \ldots, N \\
\hat{i}_N &= \arg \max_{1 \le k \le L} \delta_N[k] \\
\hat{i}_n &= \psi_{n+1}[\hat{i}_{n+1}], \quad n = N-1, \ldots, 1.
\end{aligned} \tag{1.3}$$

In order to find the most probable state at each time instant, a popular forward/backward algorithm is used [58]. In the forward/backward algorithm, the state probabilities $\{P[I_n = l|\underline{Z} = \underline{z}]\}$ are calculated. The state probabilities can be rewritten as

$$\begin{aligned}
P[I_n = l|\underline{Z} = \underline{z}] &= \frac{P[I_n = l, \underline{Z} = \underline{z}]}{P[\underline{Z} = \underline{z}]} \\
&= \frac{P[I_n = l, \underline{Z} = \underline{z}]}{\sum_{m=1}^{L} \mathrm{P[I_n = m, \underline{Z} = \underline{z}]}}.
\end{aligned} \tag{1.4}$$

By Markov assumption, the numerator can be rewritten as

$$P[I_n = l, \underline{Z} = \underline{z}] = P[z_1, z_2, \ldots, z_n, I_n = l]P[z_{n+1}, z_{n+2}, \ldots, z_N|I_n = l]. \tag{1.5}$$

We define the first term as 'forward' probabilities

$$\alpha_n[l] \equiv P[z_1, z_2, \ldots, z_n, I_n = l], \tag{1.6}$$

and the second term as 'backward' probabilities

$$\beta_n[l] \equiv P[z_{n+1}, z_{n+2}, \ldots, z_N | I_n = l]. \tag{1.7}$$

Then (1.4) can be rewritten in terms of α and β as:

$$P[I_n = l | \underline{Z} = \underline{z}] = \frac{\alpha_n[l]\beta_n[l]}{\sum_{m=1}^{L} \alpha_n[m]\beta_n[m]}. \tag{1.8}$$

Forward and backward probabilities can be recursively computed as follows [58]:

$$\begin{aligned}
\alpha_1[l] &= P[I_1 = l]P[Z_1 = z_1 | I_1 = l], \\
&\quad l = 1, \ldots, L \\
\alpha_n[l] &= \left[\sum_{k=1}^{L} \alpha_{n-1}[k]P[I_n = l | I_{n-1} = k]\right] \cdot P[Z_n = z_n | I_n = l], \\
&\quad l = 1, \ldots, L, \quad n = 2, \ldots, N \\
\beta_N[l] &= 1, \quad l = 1, \ldots, L \\
\beta_n[l] &= \sum_{k=1}^{L} P[I_{n+1} = k | I_n = l]P[Z_{n+1} = z_{n+1} | I_{n+1} = k]\beta_{n+1}[k], \\
&\quad l = 1, \ldots, L, \quad n = N-1, \ldots, 1.
\end{aligned} \tag{1.9}$$

Note that in the above algorithm, both α and β consist of the sum of a large number of terms, each of the form of product of terms that are generally significantly less than 1. It can be seen that as n starts to get big, each term starts to head exponentially to zero. The dynamic range of α and β will exceed the precision range of any machine. Hence a scaling procedure is required to avoid numerical error [58].

The above algorithm gives the state probabilities $\{P[I_n = l|\underline{Z} = \underline{z}]\}$. The most probable state at time n, $\hat{i}_n$, can be found by finding $arg \max_{i_n} P[I_n = i_n|\underline{Z} = \underline{z}]$.

1.2.4 Joint Source-Channel Decoding Techniques

In some communication systems, the transmitted sequence, $\underline{I}$, can be thought of as a state sequence of a Markov model. After a noisy channel, the receiver observes a noisy sequence. The receiver does not have access to the transmitted sequence. The states are *hidden* from the receiver. Thus, the decoding at the receiver can be modelled as an HMM problem [49]. The observed output, $\underline{Z}$, is a noisy received sequence. The receiver has to estimate the transmitted sequence. There are two basic types of decoder that we used in this thesis (chapter 4 and 5). One is a maximum *a posteriori* probability decoder [56],[53] and the other is a minimum mean squared error decoder [27],[60],[41].

1.2.4.1 Maximum a Posteriori Probability Decoder

A maximum *a posteriori* probability (MAP) decoder decodes the received message by trying to maximize either the sequence *a posteriori* probability, $P[\underline{I} = \underline{i}|\underline{Z} = \underline{z}]$ [56],or, individually, the symbol *a posteriori* probability, $P[I_n = i_n|\underline{Z} = \underline{z}]$ [53]. The former is the *sequence MAP* decoder, with the latter known as *symbol MAP* decoding.

A symbol-MAP decoder uses the well known forward/backward (F/B) algorithm introduced in (1.8) to calculate the *a posteriori* probability of the states at each time instant. A decoder then chooses the maximum *a posteriori* index at each time instant. This algorithm minimizes the symbol error probability. For a channel decoder, particularly for decoding of a convolutional code, the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm

[4], which is based on the forward/backward algorithm, is a well-known symbol-MAP decoder.

A sequence-MAP decoder, on the other hand, tries to minimize the sequence error probability by selecting the best state sequence (path) along the trellis. In other words, the decoder maximizes the *a posteriori* probability $P[\underline{I} = \underline{i}|\underline{Z} = \underline{z}]$. This is done by the dynamic programming algorithm, introduced in (1.3).

At a very low channel noise, both MAP decoders perform equally, with sequence-MAP having less computational complexity. At high channel noise, symbol-MAP has been found to experimentally outperform sequence-MAP by as much as 2 dB [49]. The advantage of a sequence-MAP decoder is that the decoded output is guaranteed to conform to the constraint of the encoder (e.g. this may be important in JSC decoding if trellis-coded quantization (TCQ) is used).

1.2.4.2 Minimum Mean Squared Error Decoder

A minimum mean squared error (MMSE) source-channel decoder first computes the *a posteriori* probability distribution for the set of source symbols; then a conditional mean estimator is applied to generate reconstruction values. This is the optimum decoder for the squared error cost. Because the reconstruction values are the average of the values in the codebook, they effectively use an infinite reconstruction "alphabet", unlike MAP detector-based decoders. Thus, the performance of the MMSE decoder is higher than that of the MAP decoder.

Consider a sequence of reproductions, $\underline{y} = (y_1, y_2, \ldots, y_N), y_n \in \mathcal{R}$, which the decoder uses to approximate the source sequence $\underline{x} = (x_1, x_2, \ldots, x_N)$. The distortion

incurred by $\underline{y}$ is $D(\underline{x},\underline{y}) = \sum_{n=1}^{N} d(x_n, y_n)$, where $d(x,y) = (x-y)^2$. The quantity the authors in [49] seek to minimize over $\underline{\hat{y}}$ is the *expected* distortion given a sequence of received indices, i.e.

$$E[D(\underline{x},\underline{y})|\underline{Z}=\underline{z}] = \frac{\sum_{\underline{i}} \left(\sum_{n=1}^{N} E[d(x_n,y_n)|\underline{I}=\underline{i}] \right) P[\underline{Z}=\underline{z}|\underline{I}=\underline{i}]P[\underline{I}=\underline{i}]}{\sum_{\underline{i}} P[\underline{Z}=\underline{z}|\underline{I}=\underline{i}]P[\underline{I}=\underline{i}]}. \tag{1.10}$$

It is shown in [49] that this minimization is not practical. An alternative solution is to use $E[d(x_n,y_n)|I_n=i_n]$ in (1.10) to approximately replace $E[d(x_n,y_n)|\underline{I}=\underline{i}]$. Because of this approximation, this decoder is called a sequence-based approximate MMSE decoder (SAMMSE). The approximation leads to minimizing

$$\hat{D} = \sum_{n=1}^{N} \sum_{l=1}^{L} \|E[x_n|I_n=l] - y_n\|^2 P[I_n=l|\underline{Z}=\underline{z}], \tag{1.11}$$

where,

$$P[i_n=l|\underline{Z}=\underline{z}] = \frac{\sum_{\underline{i}:i_n=l} P[\underline{Z}=\underline{z}|\underline{I}=\underline{i}]P[\underline{I}=\underline{i}]}{\sum_{\underline{i}} P[\underline{Z}=\underline{z}|\underline{I}=\underline{i}]P[\underline{I}=\underline{i}]}. \tag{1.12}$$

The decoder then selects the reconstruction levels according to the centroid rule [49]:

$$y_n = \sum_{l=1}^{L} E[x_n|I_n=l]P[I_n=l|\underline{Z}=\underline{z}], \quad \forall n. \tag{1.13}$$

Note that in (1.13), the probabilities $\{P[I_n=l|\underline{Z}=\underline{z}]\}$ are needed. The well-known foward/backward algorithm introduced earlier can be used to calculate these probabilities.

1.2.5 Stochastic Modeling of Images

In chapter 6, image segmentation problems are studied. Markov random fields (MRFs) theory is a branch of probability theory for analyzing the spatial and/or contextual dependencies of physical phenomena [38]. Let $\underline{I} = \{I_{(1,1)}, \ldots, I_{(X,Y)}\}$ be a family of random variables, with its realization given by $\{i_{(1,1)}, \ldots, i_{(X,Y)}\}$, defined on the set S in which each random variables $I_{(x,y)}$ takes a value $i_{(x,y)}$ in $\mathcal{L}$. The family $\underline{I}$ is called a random field. For a discrete label set $\mathcal{L}$, the joint probability is $P[\underline{I} = \underline{i}]$. F is said to be a Markov random field on S with respect to a neighborhood system N if and only if:

$$P[\underline{i}] \geq 0, \tag{1.14}$$

$$P[i_{(x,y)} | i_{S-(x,y)}] = P[i_{(x,y)} | N_{(x,y)}]. \tag{1.15}$$

The above equations state that the label at site i depends on the label of neighboring sites, $N_{(x,y)}$. The second order neighborhood of $i_{(x,y)}$ is $i_{(x-1,y-1)}$,$i_{(x-1,y)}$,$i_{(x-1,y+1)}$,$i_{(x,y-1)}$,$i_{(x,y+1)}$, $i_{(x+1,y-1)}$,$i_{(x+1,y)}$ and $i_{(x+1,y+1)}$.

A set of random variables $\underline{I}$ is a Gibbs random field (GRF) on S with respect to N if and only if its configurations obey a Gibbs distribution [38], which takes the following form:

$$P[\underline{i}] = \frac{\exp\left(\frac{-1}{T} U(\underline{i})\right)}{\sum\limits_{\underline{i} \in \underline{I}} \exp\left(\frac{-1}{T} U(\underline{i})\right)}, \tag{1.16}$$

where $U(\underline{i})$ is the energy function given by

$$U(\underline{i}) = \sum_{c \in C} V_c(\underline{i}), \tag{1.17}$$

where V_c is a clique potential. A clique c for (S, N) is defined as a subset of sites (one or a number of neighbors) in S. If we consider cliques of size up to 2, we have

$$U(\underline{i}) = \sum_{(x,y)\in S} V_1(i_{(x,y)}) + \sum_{(x,y)\in S} \sum_{(l,m)\in N_{(x,y)}} V_2(i_{(x,y)}, i_{(l,m)}). \tag{1.18}$$

Using the above expression, we can express a conditional probability as

$$P[i_{(x,y)}|N_{(x,y)}] = \frac{\exp\left(-\frac{1}{T}[V_1(i_{(x,y)}) + \sum_{(l,m)\in N_{(x,y)}} V_2(i_{(x,y)}, i_{(l,m)})]\right)}{\sum_{i_{(x,y)'}\in L} \exp\left(-\frac{1}{T}[V_1(i_{(x,y)'}) + \sum_{(l,m)\in N_{(x,y)}} V_2(i_{(x,y)'}, i_{(l,m)})]\right)}. \tag{1.19}$$

In this thesis, we will use the second order Multi Level Logistic (MLL) model [9], where only cliques of up to size 2 are used. For this type of model, the clique potentials are defined as

$$V_1(i_{(x,y)}) = \gamma_l, \tag{1.20}$$

$$V_2(i_{(x,y)}, i_{(l,m)}) = \begin{cases} \mu & if\ i_{(x,y)} = i_{(l,m)} \\ 0 & otherwise \end{cases}$$

where γ_l is the prior probability of label l, and μ is a bonding term.

In chapter 6, we try to maximize $P[\underline{i}|\underline{z}]$, where $\underline{z}$ are noisy observations. $P[\underline{i}|\underline{z}]$ has the following form:

$$\begin{aligned} P[\underline{i}|\underline{z}] &= \frac{P[\underline{z}|\underline{i}]P[\underline{i}]}{K} \\ &= \frac{1}{K}\exp\left\{\frac{1}{T}\sum_{(x,y)\in\mathcal{S}} [\log f(z(x,y)|i(x,y)) + \gamma_{i(x,y)}\right. \end{aligned}$$

$$+\frac{\mu}{2}\sum_{(l,m)\in\mathcal{N}_{(x,y)}}\delta(i(l,m)-i(x,y))]\Big\}, \tag{1.21}$$

where K a normalized constant. To maximize (1.21) is equivalent to minimize the energy function,

$$\begin{aligned} U(\underline{i}|\underline{z}) &= -\sum_{(x,y)\in\mathcal{S}}[\log f(z(x,y)|i(x,y))+\gamma_{i(x,y)} \\ &\quad +\frac{\mu}{2}\sum_{(l,m)\in\mathcal{N}_{(x,y)}}\delta(i(l,m)-i(x,y))]. \end{aligned} \tag{1.22}$$

This energy function will be used in chapter 6.

1.3 Thesis Contributions

1. We develop an iterative hillclimbing algorithm for conditional entropy-constrained encoding that builds on and greatly extends earlier work in [39]. This method allows capturing high order statistical dependencies to substantially improve encoder performance [12]. This technique is demonstrated in a transform image coding context.

2. (a) We extend the above iterative encoding for both the vertical and horizontal dimensions (row and column) to improve on the solution.

   (b) We further extend the iterative encoding algorithm to work along a *third* frequency axis, thus obtaining a method for encoding hyperspectral images.

3. We realized the proposed iterative hillclimbing algorithm can be applied to general constrained optimization problems. For example, noisy image decoding problems, and image segmentation problems.

4. The iterative *encoding* technique is an iterative assignment algorithm that is directly transferable to address joint source-channel *decoding*, based on high-order models of statistical source dependency. We modify this method for image decoding over noisy channels.

5. We investigate the use of Turbo decoding concepts to achieve improved iterative JSC decoding for systems consisting of source coders in tandem with convolutional channel coders. Although we do not consider the redundancies between (transform coefficient) sources, as in our earlier hillclimbing work, the results show a considerable improvement over non-iterative decoding techniques.

6. The iterative hillclimbing algorithm was applied to image segmentation applications in [6]. While the energy function is comparable with the results from mean-field annealing [31], the classification accuracy is inferior. This motivated us to consider an *alternative* to iterative hillclimbing sweeps over the image. Instead of dynamic programming, we investigated a method based on forward/backward algorithm sweeps over the image. Chapter 6 provides an initial, preliminary investigation of this idea, applied in the image segmentation context. Since this initial investigation, this idea has been more substantially developed [52],[51],[50]. In particular, it has been shown that:

   (a) an algorithm based on forward/backward sweeps is an extension of mean field annealing techniques, modelling dependencies between pmfs over the label *sequence* for neighboring rows.

   (b) This method descends in a free energy cost function.

Since the hillclimbing method raised the possibility of an alternative method with forward/backward sweeps replacing dynamic programming sweeps, we include a preliminary heuristic development of this approach in chapter 6 of this thesis. This is mainly given as a bridge to future work emanating from and building on this thesis. However, the full mathematical and algorithmic development and experimental evaluation of this alternative (forward/backward-based) approach is given in [50]. This approach is currently being extended for JSC decoding [51] as well as other problems.

Chapter 2

Iterative Hillclimbing Algorithm for Joint Encoding of Image Transform Coefficients

2.1 Introduction

The basic function of the source encoder is to remove redundancy from the source to reduce the required number of transmitted bits. Shannon [62] states that the minimum required bits to represent a source is equal to the source entropy. However, it is very difficult to build such an encoder with reasonable complexity in practice. Many source encoders have been proposed to attempt to get close to the source entropy with manageable computational complexity. Another approach is to design a lossy encoder. In a lossy encoder, the number of bits used to represent a source is less than the source entropy; thus distortion is unavoidable. In designing a lossy coder, a trade off between rate (number of bits) and distortion has to be optimized. Entropy-constrained quantization encoding techniques, e.g. [44], [22], seek to provide the best possible rate-distortion tradeoff by explicitly accounting for variable length codewords when choosing the quantization level to be used for a source sample.

For image encoding, usually a linear transform is applied to a small segment of the image to allow use of low-complexity scalar quantization, without a substantial sacrifice in rate-distortion performance. One can think of this as a low complexity vector

quantization. This creates transform blocks of the image. These blocks have high redundancy within them; thus it is possible for the encoder to exploit this redundancy. Most source encoders capture dependency either *within* transform blocks or *between* blocks. The well-known *intra* block encoders are the JPEG standard encoder and hierarchical wavelet encoders. Entropy-constrained trellis coded quantization (ECTCQ) [22],[46], when applied separately to each coefficient source, is one example of an *inter* block encoder. However, intra-block encoding and inter-block encoding are complementary, which motivates a new encoding scheme based on both paradigms. One such approach is *conditional* ECTCQ (CECTCQ) of block-transformed images, with conditioning used to capture both intra-block *and* inter-block redundancy. Conditional entropy-constrained (CEC) encoding has been effectively applied in several coding contexts [34],[14].

It will be shown later in this chapter that an optimal CEC encoder has intractable computational complexity. Thus, another method of encoding is needed. A standard greedy encoder is normally used as an alternative to reduce computational complexity to a manageable level by sacrificing encoding performance. It is known that the solution obtained by a standard greedy encoding is suboptimal, and cannot be claimed to be even locally optimal. In this chapter, a new iterative optimization technique is proposed to improve encoding performance over a standard greedy encoder, yet with a manageable increase in complexity.

In a standard greedy CEC encoder, the image is encoded row-by-row. Each encoding step searches for the optimal solution for a current row, given the fixed solution at the previous row. This is a step-wise optimization, which terminates after one pass through the image. However, this is not guaranteed to yield a globally optimal solution,

nor even a locally optimal solution. Moreover, because of the nature of the encoding technique (i.e. source/row-sequential), "revisiting" of row encoding choices to further reduce the cost function may be possible. This "revisiting" is a basic idea of this chapter.

The method we propose will build on and greatly extend earlier work in [39], where the authors revisit the encoded indices one sample at a time. The proposed encoder in this chapter iteratively searches for a solution over a *row sequence* of samples.

The next section shows some works that are related to this chapter. Section 2.3 will review the standard greedy encoding algorithm in detail. In section 2.3.2, a new iterative hillclimbing encoding technique is described in detail. In section 2.4, an extension to this iterative encoder is studied. Experiments and simulation procedures are given in section 2.5. Section 2.6 shows the experimental results. Finally, a discussion and conclusion are given in section 2.7.

2.2 Related Works

Several authors have proposed techniques that improve upon standard greedy search while retaining manageable complexity. [15] proposed a soft input/soft output iterative optimization, alternately applied along the image rows and columns, based on "turbo decoding" concepts. Their approach was demonstrated to outperform a greedy search for image halftoning and entropy-constrained encoding. Our method is similar to [15] in its general applicability and in that we also suggest iterative row-column optimizations. One advantage of our method is its theoretically guaranteed improvement with each iteration, i.e. our method performs hillclimbing on the cost surface. By initializing with the greedy solution, we guarantee gains over greedy search, and additional gains

with each iteration. No such theoretical guarantees are provided in [15]. However, a potential advantage of [15] is its propagation of soft decoding information – by contrast, our method iterates in the space of hard index assignments. We also note [47], where a rate-distortion optimized "bit flipping" algorithm was developed for lossy/lossless compression of bilevel images, based on arithmetic coding. Similar to our work, this approach performs multiple iterations (passes) over the image, with each pass improving the overall objective function. However, in [47] bit flipping is considered one pixel at a time, whereas we make *jointly optimal* index assignments for entire rows/columns of the image at each step. From our experience, iterative optimization based on joint assignments (e.g. one row at a time) typically achieves better solutions than that based on choosing one pixel at a time. Finally, we note [40], where groups of rows were jointly decoded in a noisy channel context, but without multiple image passes.

2.3 Formulation

In this section, formulations will be given and described. We first introduce a standard greedy encoding algorithm, which is a basic encoding technique that we will use to compare the results with. Then, a new kind of iterative encoding algorithm will be introduced.

For transform coding of images, the image is first divided into blocks (e.g. 8 $\times$ 8 pixels). A linear transform such as 2-D DCT is applied to each block. Then grouping together all coefficients at a common spatial frequency, we effectively create a source at each spatial frequency. Thus, the number of sources is given by the block size. Each source is represented as a 1-D sequence by using a fixed scan order for

visiting the transformed blocks, e.g. row-by-row. Moreover, the sources are themselves ordered according to increasing spatial frequency, e.g. via a zig-zag scan. This process is summarized in Figure 2.1.

Consider a collection of source sequences $\{\underline{s_1}, \underline{s_2}, \ldots, \underline{s_M}\}$, where $\underline{s_m} \equiv (s_{m1}, s_{m2}, \ldots, s_{mN})$, $s_{mn} \in \mathcal{R}$ is a sequence of samples from the mth source. The collection of sources can be organized as a 2D array as in Figure 2.3. These samples will be encoded (quantized) by the encoder. Thus the quantization index used for sample s_{mn} will be denoted by i_{mn}, with the associated quantization level given by the discrete mapping $q_m(i_{mn})$. Note that the subscript m means each source sequence has a distinct quantizer. Furthermore, we will need to represent the *sequence* of encoded indices for each source, i.e. $\underline{i_m} \equiv (i_{m1}, i_{m2}, \ldots, i_{mN})$. Figure 2.4 shows the 2-dimensional array of encoded indices.

2.3.1 Standard Greedy Conditional Entropy-Constrained Encoding

A conditional entropy-constrained (CEC) encoder uses a conditional probability model to express the statistical dependency between samples. In an image coding application, a conditional probability model is used to represent the dependency between encoded transform coefficients (i.e. dependency between source sequences), and the dependency between encoded indices within the same source. For simplicity, a second-order Markov probability model is assumed, i.e. the model will be based on the probabilities $\{Prob[i_{mn}|i_{m,n-1}, i_{m-1,n}]\}$, indicated by arrows in Figure 2.4.

The objective of the CEC encoder is to select an array of quantization indices to optimize (minimize) the Lagrangian cost function $J \equiv D + \lambda R$, with D the quantizer

distortion and R the source coding description length in bits. This is a basic rate-distortion cost. We find it useful to represent J as a sum of *source-wise* costs, i.e.

$$J = (D_1(\underline{i_1}) + \lambda R_1(\underline{i_1})) + \sum_{m=2}^{M} (D_m(\underline{i_m}) + \lambda R_m(\underline{i_m}, \underline{i_{m-1}})). \tag{2.1}$$

Here, $D_m(\underline{i_m}) = \sum_{n=1}^{N} d(s_{mn}, q(i_{mn}))$ with $d(\cdot,\cdot)$ a specified scalar distortion measure, $R_1(\underline{i_1}) = l(i_{11}) + \sum_{n=2}^{N} l(i_{1n}; i_{1,n-1})$, and $R_m(\underline{i_m}, \underline{i_{m-1}}) = l(i_{m1}; i_{m-1,1}) + \sum_{n=2}^{N} l(i_{mn}; i_{m-1,n}, i_{m,n-1})$, $m \geq 2$.

In (2.1), notation in $R_m(\cdot)$ involving $\underline{i_m}$ and $\underline{i_{m-1}}$ emphasizes that the bit length associated with the mth source ($m \geq 2$) depends on the encoding choices for both $\underline{s_m}$ and $\underline{s_{m-1}}$. It is this dependence which makes optimal CEC encoding only achievable by an exhaustive search over all possible ($\prod_{m=1}^{M} (L_m)^N$) image encodings. Since this exhaustive encoder is quite infeasible, a practical, albeit *greedy* alternative must typically be used, e.g. [36].

The standard greedy encoding algorithm is a sub-optimal encoding algorithm since it does not search over all possible solutions. However, because it requires less computational complexity, it is a popular encoding technique. Given the encoded samples for the previous sequence (row), the algorithm searches for an optimal solution for the current row. Given this solution, the algorithm searches for a solution for the next row. This repeats until all sequences (rows) are encoded.

Standard Greedy CEC Encoding:

1. Use dynamic programming (Viterbi algorithm) to minimize

   $D_1(\underline{i_1}) + \lambda R_1(\underline{i_1})$ over $\underline{i_1}$.

   Let $\underline{i_1^{(0)}}$ denote the solution.

2. For m=2 to M

   - Use dynamic programming 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

While this procedure is *step-wise* optimal in the sense of choosing the best sequence $\underline{i_m}$ given fixed $\underline{i_{m-1}^{(0)}}$, it does not guarantee even a *locally* optimal collection $\{\underline{i_m},\ m = 1, \ldots, M\}$. In particular, after determining $\underline{i_m^{(0)}}$ given $\underline{i_{m-1}^{(0)}}$, it is quite possible that some index $i_{m-1,n}$ could be re-chosen to reduce the Lagrangian cost. In fact, such "revisiting" of previously made encoding choices is what forms the basis of our new paradigm. Note that if "revisiting" is attempted for a standard greedy encoding scheme, the encoder will again first minimize $D_1(\underline{i_1}) + \lambda R_1(\underline{i_1})$ over $\underline{i_1}$. The solution $\underline{i_1^{(1)}}$ is exactly $\underline{i_1^{(0)}}$. In other words, there is no benefit to revisiting encoding choices within standard greedy encoding. Thus, we propose a modification to standard greedy encoding which is described next.

2.3.2 Iterative Hillclimbing Conditional Entropy-Constrained Encoding

The previous section shows how a standard greedy encoder works. The encoding performance can be further improved by re-adjusting previously encoded indices.

However, if "revisiting" is performed by a standard greedy encoder, i.e. after encoding last row, the encoder goes back to the first row and starts encoding again, the solution obtained from this second iteration will be exactly the same as the one obtained from the first iteration.

Crucial to our method is a new "source-wise" cost to be minimized in choosing $\underline{i_m}, m < M$. Note in particular that, in (2.1), the sequence $\underline{i_m}$ affects $D_m(\underline{i_m})$, $R_m(\underline{i_m}, \underline{i_{m-1}})$, *and* $R_{m+1}(\underline{i_{m+1}}, \underline{i_m})$. Thus, suppose we have already used the greedy encoding technique to determine $\{\underline{i_m^{(0)}}, m = 1, \ldots, M\}$. Re-selecting $\underline{i_m}$ to minimize $D_m(\underline{i_m}) + \lambda R_m(\underline{i_m}, \underline{i_{m-1}^{(0)}})$ will simply lead again to the solution $\underline{i_m^{(0)}}$. However, suppose instead that we now minimize $D_m(\underline{i_m}) + \lambda R_m(\underline{i_m}, \underline{i_{m-1}^{(0)}}), +\lambda R_{m+1}(\underline{i_{m+1}^{(0)}}, \underline{i_m})$. This minimization can *also* be achieved via dynamic programming, simply by adding the terms $\lambda l(i_{m+1,n}^{(0)}; i_{m,n}, i_{m+1,n-1}^{(0)})$ to the branch metrics used in the dynamic programming algorithm for minimizing $D_m(\underline{i_m}) + \lambda R_m(\underline{i_m}, \underline{i_{m-1}^{(0)}})$. Since this new "source-wise" cost $(D_m(\underline{i_m}) + \lambda R_m(\underline{i_m}, \underline{i_{m-1}^{(0)}}) + \lambda R_{m+1}(\underline{i_{m+1}^{(0)}}, \underline{i_m}))$ is composed of $(D_m(\underline{i_m}) + \lambda R_m(\underline{i_m}, \underline{i_{m-1}^{(0)}}))$ plus an *additional* term from J, and since dynamic programming is guaranteed to find the global minimum solution, the resulting sequence $\underline{i_m^{(1)}}$ *must* be at least as good in the sense of J as the initial one, $\underline{i_m^{(0)}}$. In practice, such minimizations are likely to provide at least some reduction in J. Thus, we suggest such minimizations as the basis for the following iterative encoding algorithm. (A more detailed proof that our approach is a hillclimbing algorithm is given in the appendix.)

Iterative, Hillclimbing CEC Encoding

1. Implement the greedy encoding to get

   $\{\underline{i_m^{(0)}}, m = 1, \ldots, M\}$.

2. Set $t = 0$

3. Do {

   - $t \leftarrow t + 1$
   - Using dynamic programming, choose $\underline{i_1^{(t)}}$ that minimizes

     $D_1(\underline{i_1^{(t)}}) + \lambda R_1(\underline{i_1^{(t)}}) + \lambda R_2(\underline{i_2^{(t-1)}}, \underline{i_1^{(t)}})$

   - For m=2 to M-1

     – Using dynamic programming, choose $\underline{i_m^{(t)}}$ that minimizes

       $D_m(\underline{i_m^{(t)}}) + \lambda R_m(\underline{i_m^{(t)}}, \underline{i_{m-1}^{(t)}}) + \lambda R_{m+1}(\underline{i_{m+1}^{(t-1)}}, \underline{i_m^{(t)}})$

     End

   - Using dynamic programming, choose $\underline{i_M^{(t)}}$ that minimizes

     $D_M(\underline{i_M^{(t)}}) + \lambda R_M(\underline{i_M^{(t)}}, \underline{i_{M-1}^{(t)}})$

   } Until (There are no encoding changes) OR (a convergence criterion is met)

Note that the above algorithm is an extension of an iterative encoding technique first proposed in [39] for product code VQ, e.g. for mean-gain-shape VQ. The proposed method extends this technique by:

1. jointly optimizing over a much larger encoding space (a full image).

2. taking optimization steps that are substantially less greedy – in [39], one encoder index was re-optimized at a time (e.g. a single mean, gain, or shape index) with all the remaining ones fixed, whereas in our approach a long *sequence* of indices is re-optimized together, given the remaining ones fixed.

This iterative CEC encoding guarantees a sequence of solutions that are non-increasing in the cost function (i.e. it is a hillclimbing algorithm). The proof sketch is given in the appendix. But it does not guarantees a locally optimal solution. We can show that we can still improve the performance by doing column-wise encoding. Our encoder searches for the optimal solution row-by-row. It is possible for the encoder to also search for the optimal solution column-by-column. This method is shown in the next section.

2.4 Extension to Hillclimbing Iterative Encoder

Recall from Figure 2.4, transform coefficients can be arranged and displayed as a two-dimensional array. The encoder sequentially uses dynamic programming to search for optimal path for each source. We will call this method a *row-wise encoding pattern.* Technically, one can think that the encoder searches *across* transform blocks for the solution. Instead of row-wise encoding, one can also try to search for the solution *column-wise.* In this case, solutions are searched *within* each transform block or based on a pre-defined scan order for the transform coefficients. We will call this method a *column-wise encoding.*

By fixing $(i_{1,n-1}, i_{2,n-1}, \ldots, i_{M,n-1})$ and $(i_{1,n+1}, i_{2,n+1}, \ldots, i_{M,n+1})$, we can encode the sequence $(i_{1,n}, i_{2,n}, \ldots, i_{M,n})$ by using similar iterative algorithm as in

the previous section. We can then "sweep" in this fashion along columns of the source "matrix" (Figure 2.4). The encoding cost for column n, $1 \leq n < N$ is $\sum_{m=1}^{M} d(s_{mn}, q_m(i_{mn}; i^{(0)}_{m,n-1})) + (\lambda l(i_{1n}; i^{(0)}_{1,n-1}) + \lambda \sum_{m=2}^{M} l(i_{mn}; i_{m-1,n}, i^{(0)}_{m,n-1})) + (\lambda l(i^{(0)}_{1,n+1}; i_{1,n}) + \lambda \sum_{m=2}^{M} l(i^{(0)}_{m,n+1}; i_{m,n}, i^{(0)}_{m-1,n+1}))$ [1]. Each such sweep will be non-increasing in the cost function, just as the sweeps along the rows.

We can even iterate row and column sweeps to potentially gain even more performance.

Row-Column Hillclimbing-based CEC Encoding

1. Implement the greedy encoding to get $\underline{\mathbf{i}}^{(0)} = \{\underline{i^{(0)}_m}, m = 1, \ldots, M\}$.

2. Set $t = 1$

3. Do {

   - Choose $\underline{\mathbf{i}}^{(t)}$ using hillclimbing (until convergence) along the rows, starting from $\underline{\mathbf{i}}^{(t-1)}$

     $t \leftarrow t + 1$

   - Choose $\underline{\mathbf{i}}^{(t)}$ using hillclimbing (until convergence) along the columns, starting from $\underline{\mathbf{i}}^{(t-1)}$

     $t \leftarrow t + 1$

   } Until (There are no encoding changes) OR (a convergence criterion is met)

[1] This cost requires specialization for $n = N$.

It is very interesting to note that this column-wise encoding technique just shows that the solution obtained from a row-wise encoding may be improved. In fact, one can try to improve performance even more by encoding *diagonally*. There are many possible numbers of scanning patterns for encoding. For simplicity, we will focus only on row-wise and column-wise encoding.

2.5 Experiment

This section explains how the simulation is set up and how the results are obtained.

We have evaluated our method for transform coding, in comparison with greedy CECTCQ encoding and with ECTCQ (where no conditioning was used). We used 8×8 image blocks, the 2-D DCT transform, and a zig-zag scan for 1-D ordering of the sources (Figure 2.2). Probability model used in this experiment is $\{Prob[i_{mn}|i_{m,n-1}, i_{m-1,n}]\}$. Figure 2.1 shows the steps to obtain the coefficient sources.

Four-state TCQ was used on each source, with the trellis as given in the original TCQ paper [44]. The codebook size for each source was determined by a bit allocation strategy [45]. A bit allocation map for 1 bpp is shown in Figure 2.5. A training set of 8 images was used for designing the coders (the TCQ codebooks and the Huffman code tables). One Huffman code was designed for each (TCQ state, conditioning context) pair.

For ECTCQ, the standard design algorithm was used, based on set partitioning for code initialization and an entropy-constrained version of Stewart and Gray's trellis-based Lloyd algorithm [63], to refine the quantization levels and entropy codes [22],[46].

For standard greedy CECTCQ, a sequential design algorithm (paralleling the encoder's operation) was employed, with the coders for each individual source designed sequentially (in zig-zag scan order, Figure 2.2), with each in turn then used to create conditioning context for the next source coder design.

Finally, for our iterative CECTCQ coder, we also devised a design algorithm matched to the encoder's operation. First, the (just described) standard CECTCQ design was used to obtain initial coders. Then, paralleling the iterative encoder, we implemented an iterative *design* algorithm, with each source coder in turn rechosen to minimize its associated "source-wise" Lagrangian cost, with the minimization over both the quantizers and the entropy codes. Cycling through the coder designs continues until either there are no further changes or until a stopping condition is reached. Mirroring the encoder's operation, this design algorithm is non-increasing in the (training set) cost J.

The resulting curves were obtained by designing the various coders for a sequence of λ values, to sweep out a rate/distortion curve. These curves will be plotted and discussed in the results section.

2.6 Results

In this section, a discussion of results based on the experiments explained in section 2.5 is given.

Figure 2.5 shows the bit allocation map for an overall rate of 1 bit per pixel. The figure shows an 8×8 block which corresponds to a DCT transform block. Note that most

of the high frequencies coefficients (bottom-right half) have zero bit allocation. This is due to the fact that most energy in images lie in lower frequencies.

In Figure 2.8, the peak signal to noise ratio (PSNR) is plotted against the total rate, where PSNR (dB) is $10 * \log_{10} \frac{255^2}{E\{(s_n - q(i_n))^2\}}$. The results are for the 512×512 Lena image (outside training images) (Figure 2.6). We have evaluated performance for 1D source dependencies (based on zig-zag scan order, Figure 2.2). As seen from the plot, the PSNR of the iterative algorithm is always higher than the PSNR of the greedy algorithm, at the same rate. At a rate of approximately 0.15 bits per pixel, the result from the iterative algorithm is almost 1dB better than that from the greedy algorithm, and 1.2dB better than a standard entropy-constrained algorithm. However, at higher rate, the gap between the algorithms is narrower. At a rate of 1 bit per pixel, the iterative algorithm is 0.4dB better that the greedy algorithm, and 0.8dB better than the entropy-constrained algorithm.

In Figure 2.9, the peak signal to noise ratio (PSNR) is plotted against the total rate for the 512×512 Fishing Boat image (outside training images) (Figure 2.7). Again, the PSNR of the iterative algorithm is always higher than the PSNR of the greedy algorithm, at the same rate. At a rate of approximately 0.15 bits per pixel, the result from the iterative algorithm is almost 0.5dB better than that from the greedy algorithm. At a rate of 1 bit per pixel, the iterative algorithm is 0.5dB better that the greedy algorithm, and 0.8dB better than the entropy-constrained algorithm.

We have also evaluated our row-column extension. For this experiment, we again applied CEC encoding to transform coefficients, but used scalar rather than trellis quantization. The trellis structure drastically constrains the possible choices available for

optimizing along columns, given fixed rows. For column optimization (following a row optimization), the encoding choices must preserve the sequence of trellis *branches* that were chosen by the row optimization, allowing optimization only over the quantization choices *within* each branch. This greatly reduces the degrees of freedom that can be exercised by a column optimization and limits possible performance benefits. Thus, to best assess the gains achievable by row-column optimization, scalar rather than trellis quantization was used. Over the bit rate range from 0.1 to 1.1 bits per pixel, the gains of the row-column method over row-only hillclimbing were found to be between 0.1-0.2 dB. For example, at 0.5 bits/sample, the basic hillclimbing method gained 0.3 dB over greedy encoding, while the row-column extension achieved 0.5 dB over greedy encoding. This additional improvement, however, requires increased encoding complexity (convergence was achieved after 10 iterations of row-column encoding).

The gains of our method are achieved at the cost of increased complexity. To quantify this, we compare execution times. All methods were implemented in C on a Sparc5. No code optimization was performed. At 0.2 bits/sample, the greedy execution time on Lena was 2.9 seconds, while the hillclimbing execution (corresponding to 7 iterations) was 46.8 seconds. At 1 bit/sample, greedy execution was 24.2 seconds versus 334.8 seconds (6 iterations) for hillclimbing. Note that hillclimbing execution is roughly '$2 \times$ number of iterations $+ 1$' times greater than greedy execution time.

2.7 Conclusion

A new kind of iterative optimization technique was proposed on image compression applications. The algorithm iteratively encodes the sources, resulting in a non-increasing cost function. The obtained solution is guaranteed to be no worse than the solution obtained from a standard greedy encoding. However, we cannot claim the obtained solution is locally optimal. As shown in section 2.4, a column-wise encoding method can further improve the performance. It is still an open question to find the optimum encoding pattern.

The computational complexity of the proposed iterative hillclimbing algorithm is between the optimal CEC algorithm and the standard greedy algorithm. In other words, we gain rate-distortion performance with the cost of increased complexity.

The proposed iterative algorithm in this chapter was used in image encoding applications, with the objective is to assign one of a finite set of index values to each pixel site, to minimize a *particular* global cost. We realized that this kind of iterative algorithm can be applied to other general optimization applications as well. For example, a noisy decoding problem, which will be studied in chapter 4 and 5, requires an algorithm to search for the most probable transmitted indices. Chapter 6 shows how our iterative optimization idea can also be applied to image segmentation applications. The objective for both noisy decoding and image segmentation is similar to image encoding, except that the global cost for each case is different.

The following chapters will explain in detail how the same iterative optimization idea and an extension of this idea can be applied to various applications.

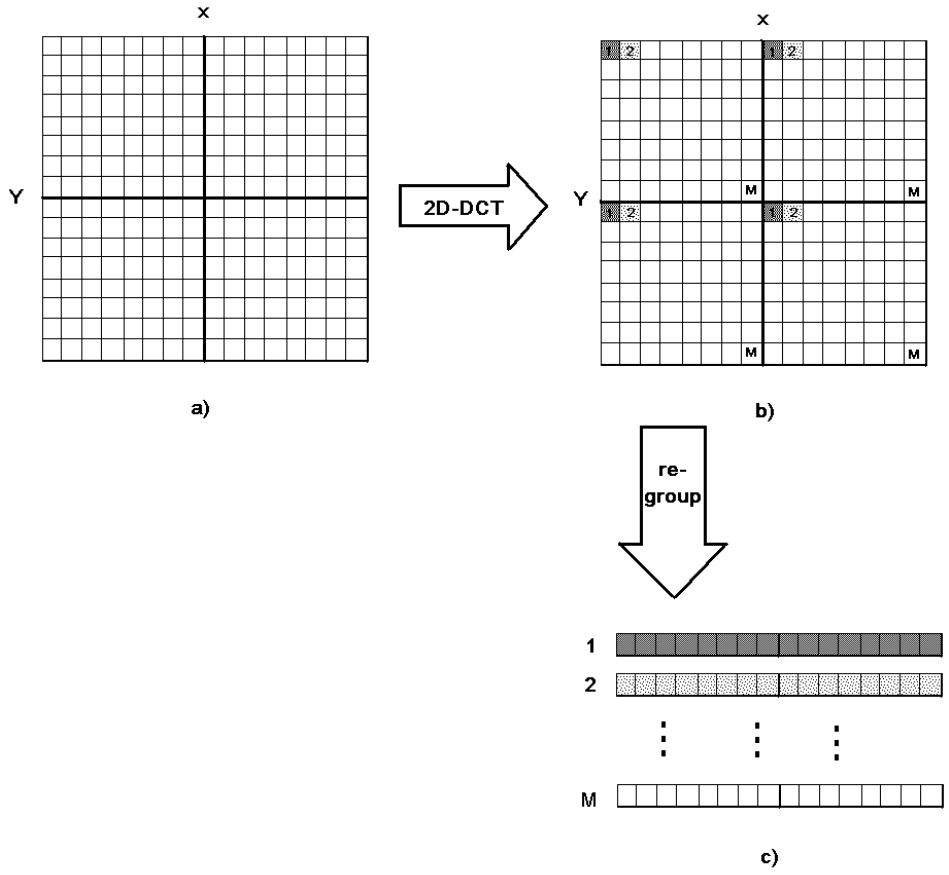

Fig. 2.1. Image transformation and sources rearrangement procedure. a) Divide image into blocks. b) After 2D-DCT, coefficients are obtained. c) Group coefficeint at the same frequency to form a sequence.

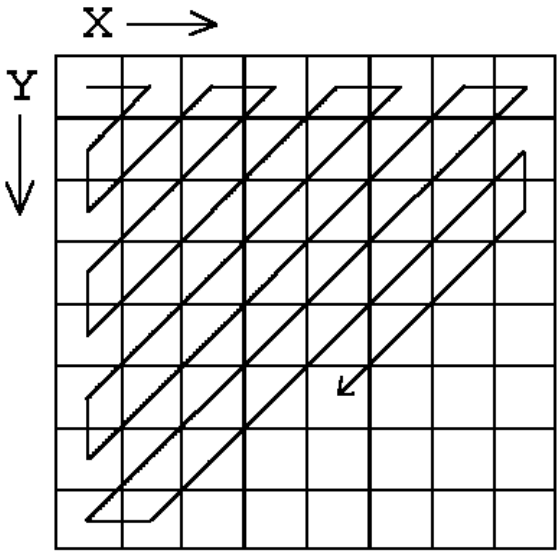

Fig. 2.2. A Zig-zag scanning pattern, resulting in a 1-D ordering of sources.

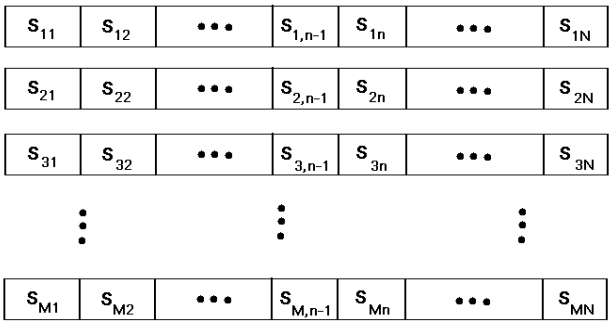

Fig. 2.3. A 2-D array of source samples.

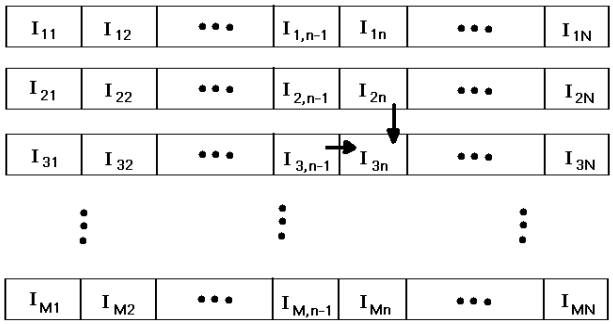

Fig. 2.4. A 2-D array of encoded indices. Arrows indicate within source and between source dependencies.

X ⟶

Y ↓

7.25	4.0	3.0	2.25	1.5	1.5	1.5	1.25
4.0	3.0	2.25	1.75	1.25	1.0	0.0	0.0
3.0	2.25	1.75	1.25	1.0	0.0	0.0	0.0
2.25	1.75	1.25	1.0	0.0	0.0	0.0	0.0
1.75	1.25	1.0	0.0	0.0	0.0	0.0	0.0
1.5	1.0	0.0	0.0	0.0	0.0	0.0	0.0
1.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.25	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig. 2.5. Result of a bit allocation on coefficient sources for 1 bit per pixel.

Fig. 2.6. 512×512 Lena image.

Fig. 2.7. 512 × 512 fishing boat image.

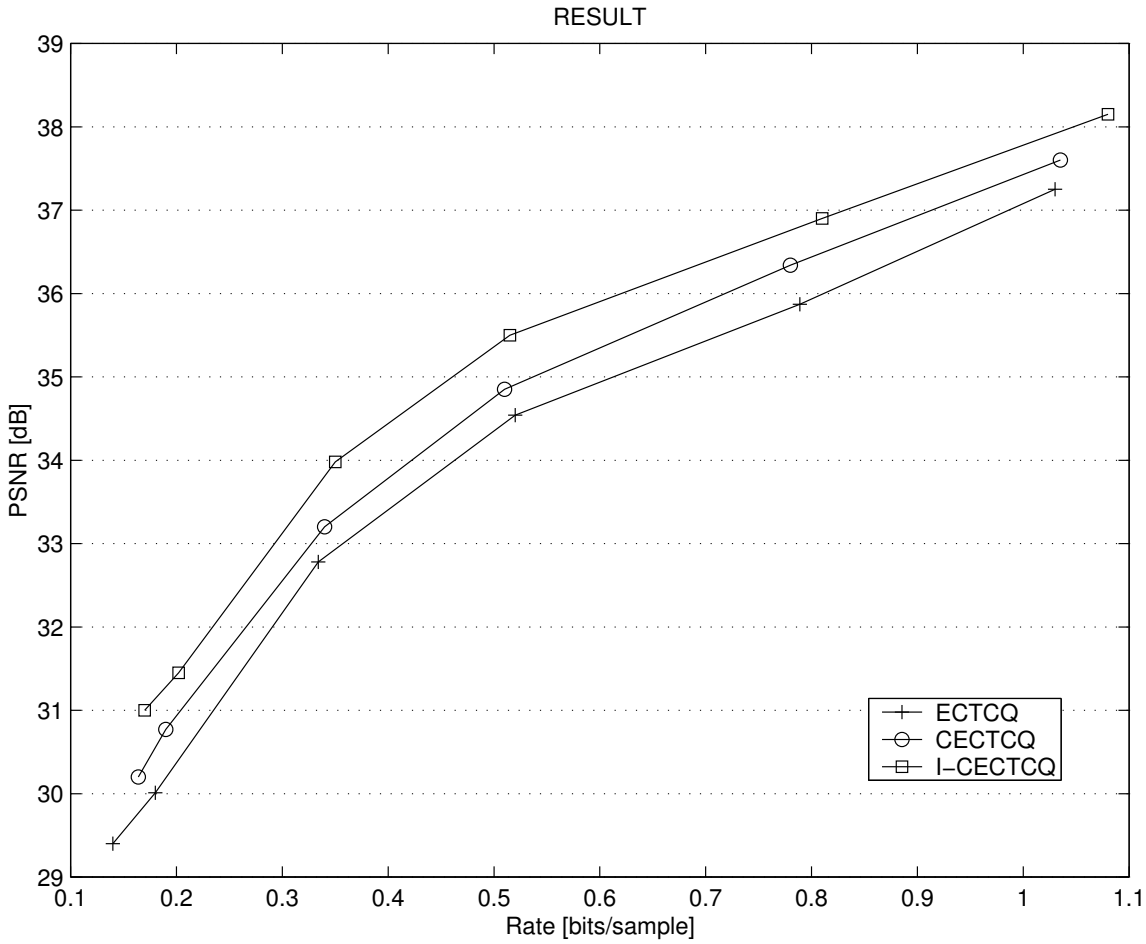

Fig. 2.8. PSNR vs Rate results on 512 $\times$ 512 Lena image.

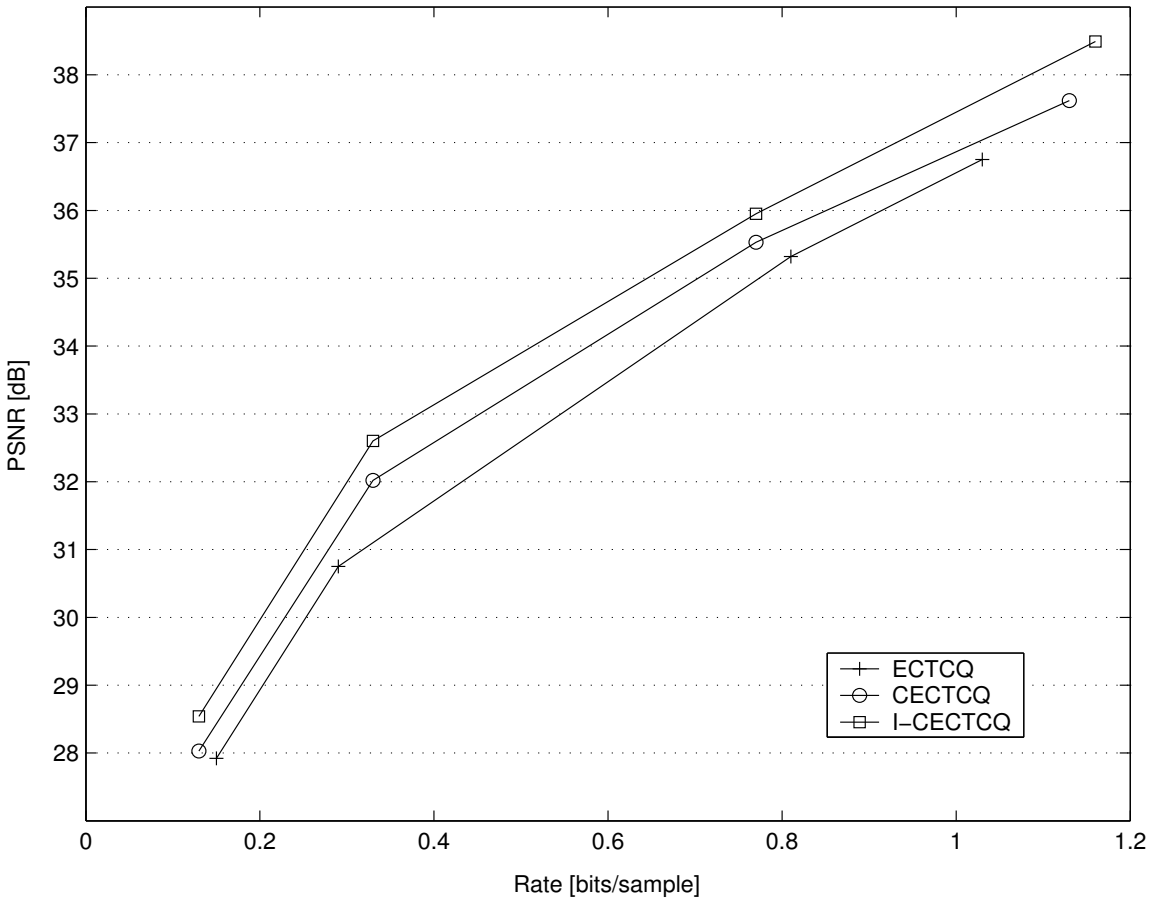

Fig. 2.9. PSNR vs Rate results on 512 × 512 Fishing Boat image.

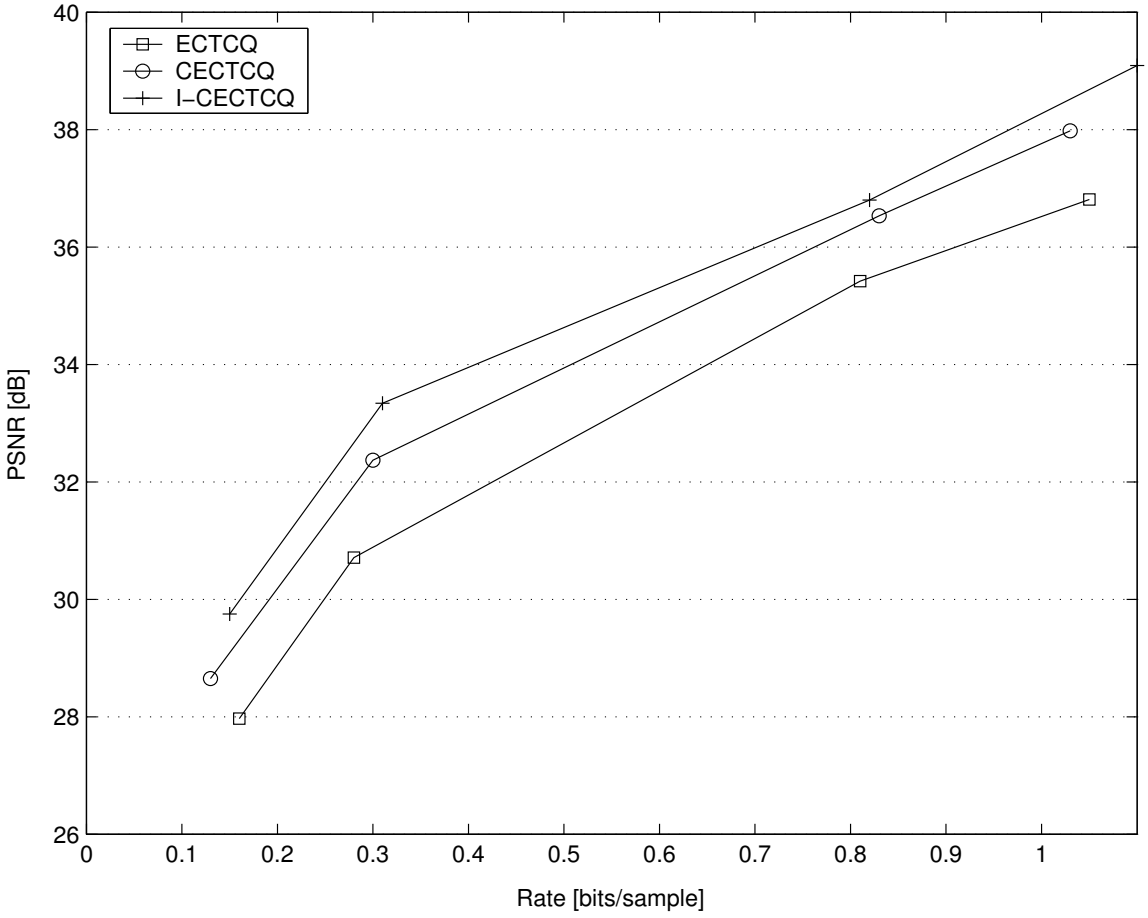

Fig. 2.10. PSNR vs Rate results on 512×512 Fishing Boat image, using 2-D conditional model.

Chapter 3

Iterative Encoding for Hyperspectral Images

3.1 Introduction

In this chapter, the iterative hillclimbing encoding technique introduced in chapter 2 will be modified and applied for hyperspectral image encoding. It will be shown in this chapter that, with a slight modification in the probability model, the iterative image encoding algorithm given in the previous chapter can be used to encode hyperspectral images.

First, hyperspectral images will be introduced in section 3.1.1. A review of many hyperspectral image encoders will be given in section 3.1.2. A new iterative hillclimbing encoding algorithm, which is based on the algorithm in chapter 2, will be introduced in section 3.3. Section 3.4 explains how the experiments were set up to obtain the results, which will be discussed in sectoin 3.5. Finally, the conclusion is given in section 3.6.

3.1.1 Hyperspectral Images

A hyperspectral image is an image that represents the same spatial area with different spectral wavelengths. Usually, hyperspectral images are taken by satellites or high-flying aircrafts with specific sensor detectors. For example, Advanced Very High Resolution Radiometers (AVHRR) are on board the NOAA -7, -9, -11 and -14 polar

orbiting satellites [1]. Each satellite has 5-channel instruments to detect spectral bandwidth ranges from 0.58-12.5 micrometers (μm) [67]. Another example is Airborne Visible InfraRed Imaging Spectrometer, (AVIRIS) [2]. Data from this system will be used for experimental purpose in this chapter.

Images taken by AVIRIS use wavelengths ranging from 400 to 2500 nanometers (nm) [3]. The AVIRIS instrument contains 224 different detectors, each with a spectral bandwidth of approximately 10 nm, allowing it to cover the entire range between 380 nm and 2500 nm. High-flying aircrafts equipped with the AVIRIS instrument produce 614 pixels for the 224 detectors each scan. Each pixel produced by the instrument covers an approximately 20 meter square area on the ground (with some overlap between pixels), thus yielding a ground swath about 11 kilometers wide. The ground data is recorded on board the instrument along with other engineering data and the readings from the AVIRIS on-board calibrator. When all of this data is processed and stored on the ground, it yields approximately 140 Megabytes (MB) for every 512 scans (or lines) of data. Each 512 line set of data is called a"scene", and corresponds to an area about 10km long on the ground.

One can imagine hyperspectral images as 3-D images, where x- and y-axis represent 2-D spatial area, and z-axis represents spectral bands. A sample of a hyperspectral image *cube* is shown in Figure 3.1. Each slice along the z-axis represents an image taken with different spectral frequency. Since each spectral band represents the same spatial area, there is a high correlation between bands. This redundancy can be exploited by

[1] http://podaac.jpl.nasa.gov/sst/.

[2] http://makalu.jpl.nasa.gov/aviris.html.

the encoder. Moreover, within the same band, there is redundancy that we exploited in chapter 2. In this chapter, the proposed encoder will capture statistical dependencies both within and across bands.

3.1.2 Hyperspectral Image Encoding

There are several works on lossy compression of multispectral and hyperspectral imagery. Some works are based on vector quantization (VQ) schemes [30], [32] [67], and [43]. [30] uses nonlinear predictive VQ to predict interband vector samples. A standard VQ is used to encode intraband samples. Thermal Infrared Multispectral Scanner (TIMS) and Calibrated Airborne Multispectral Scanner (CAMS) images are studied and encoded by spectral VQ in [32], with a compression ratio of 24:1 and 28:1, respectively. AVHRR system with VQ compression is studied in [67], with a compression ratio of 24:1. AVHRR system is also studied in [43], but a progressive VQ with the use of a single instruction multiple data machine is used instead.

Many non-VQ systems have also been introduced. [66] uses a hierarchical data compression system, which allows many levels of compression quality. [59] uses 3-D Karhunen-Loeve (KL) transformation to decorrelate spectrally, and uses adaptive discrete cosine transformation (DCT) coding technique to encode spatially. A similar system was proposed in [18],but the discrete wavelet transform (DWT) is used to spatially decorrelate the principal components. In [2], differential pulse code modulation (DPCM) is used to spectrally decorrelate the hyperspectral data (AVIRIS), while a 2-D DCT

coding is used for spatial decorrelation. Then an entropy-constrained trellis coded quantization (ECTCQ) is used to quantize transform coefficients. This is the basic system that we will compare with. Next section will discuss more on this reference system.

3.2 Hybrid DPCM/DCT and ECTCQ for Hyperspectral Image

This section explains in detail the basic encoding system used in this chapter.

Because of the nature of hyperspectral images, there are high correlations between spectral bands. A combination of a differential pulse code modulation (DPCM) and DCT can be used to decorrelate the images. Using DPCM, error images are obtained for each spectral band. These error images have lower energy than original images, therefore less quantization bits are needed to encode these error images. [2] proposed a hybrid DPCM/DCT and ECTCQ for hyperspectral image encoding. The system in Figure 3.2 is used. For each spectral band k, $\underline{\mathbf{X}}^k$,is the input image of the DPCM loop. After subtracting the input band with a predicted image, an *error* image , $\underline{\mathbf{E}}^k$, is produced.

A 2-D DCT is performed on each 8×8 block of error image. After linear transformation, similarly to previous chapter, grouping of like transform coefficients is performed. Entropy-constrained TCQ is then used to encode each coefficient source. Then, encoded indices can be transmitted or stored. A reverse procedure is performed on encoded indices in order to obtain a *decoded error* image. Specifically, an ECTCQ decoder is first used to decode the encoded sequence. After regrouping coefficient sources back to original 8×8 blocks, an inverse 2-D DCT is performed on each block to obtain a decoded error image, $\hat{\underline{\mathbf{E}}}^k$. Note that $\hat{\underline{\mathbf{E}}}^k$ is not equal to $\underline{\mathbf{E}}^k$. The difference is due to quantization noise. $\hat{\underline{\mathbf{E}}}^k$ is combined with previously predicted image to get the decoded spectral band

image, $\underline{\hat{\mathbf{X}}}^k$. $\underline{\hat{\mathbf{X}}}^k$ is then used as an input to a linear predictor to predict the next spectral band image, $\underline{\tilde{\mathbf{X}}}^{k+1/k}$. This process is a DPCM loop. The encoding process repeats for all spectral bands. This process is referred to as a greedy encoding algorithm, since the encoder encodes (optimizes) one band at a time. It might be possible to go back and readjust encoded sequences to obtain a better result.

In this chapter, the above system will be used as a reference. The difference is that a CECTCQ will be used instead of ECTCQ to capture statistical dependencies both *within* and *between* spectral bands. Also, the iterative algorithm will be applied, instead of a standard greedy algorithm. The algorithm will be explained in the next section.

3.3 Iterative Encoder for Hyperspectral Image

This section gives a mathematical formula for the proposed iterative algorithm.

After applying 2-D DCT on the error image, consider a collection of source (error) sequences for the kth band $\{\underline{e^k_1}, \underline{e^k_2}, \ldots, \underline{e^k_M}\}$, where $\underline{e^k_m} \equiv (e^k_{m1}, e^k_{m2}, \ldots, e^k_{mN})$, $e^k_{mn} \in \mathcal{R}$. Denote the quantization index used for sample e^k_{mn} by i^k_{mn}, with the associated quantization level given by the discrete mapping $q^k_m(i^k_{mn})$. Furthermore, we will need to represent the *sequence* of encoded indices for each source and for each band, i.e. $\underline{i^k_m} \equiv (i^k_{m1}, i^k_{m2}, \ldots, i^k_{mN})$. The collection of quantization indices, for all spectral bands, can be organized as a stack of images, as shown in Figure 3.3.

Similar to chapter 2, the objective of the CEC encoder is to select an array of quantization indices to minimize the Lagrangian cost function $J \equiv D + \lambda R$, where D is a distortion, and R is an encoding rate (number of bits). To utilize the iterative hillclimbing algorithm proposed in the previous chapter, J will be expressed as a sum of

source-wise costs, i.e.

$$J = D_1^1(\underline{i_1^1}) + \lambda R_1^1(\underline{i_1^1}) + \sum_{m=2}^{M} (D_m^1(\underline{i_m^1}) + \lambda R_m^1(\underline{i_m^1}, \underline{i_{m-1}^1}) + \sum_{k=1}^{K} \left[D_1^k(\underline{i_1^k}) + \lambda R_1^k(\underline{i_1^k}, \underline{i_1^{k-1}}) + \sum_{m=2}^{M} (D_m^k(\underline{i_m^k}) + \lambda R_m^k(\underline{i_m^k}, \underline{i_{m-1}^k}, \underline{i_m^{k-1}})) \right]. \tag{3.1}$$

Here, $D_m^k(\underline{i_m^k}) = \sum_{n=1}^{N} d(s_{mn}^k, q_m^k(i_{mn}^k))$ with $d(\cdot,\cdot)$ a specified scalar distortion measure, $R_m^k(\underline{i_m^k}, \underline{i_{m-1}^k}, \underline{i_m^{k-1}}) = l(i_{m1}^k; i_{m-1}^k, i_{m1}^{k-1}) + \sum_{n=2}^{N} l(i_{mn}^k; i_{m-1,n}^k, i_{m,n-1}^k, i_{mn}^{k-1})$. Note that special care must be taken when the source is at the border (i.e. the first spectral band uses $R_m^1(\underline{i_m^1}, \underline{i_{m-1}^1})$, and the first row of each spectral image uses $R_1^k(\underline{i_1^k i_1^{k-1}})$.)

From the expression of R_m^k, it is clear that a conditional probability model $Prob[i_{mn}^k | i_{m,n-1}^k, i_{m-1,n}^k, i_{mn}^{k-1}]$ is used to capture the dependencies both *within* and *between* spectral bands. The current encoded index,i_{mn}^k , depends on three other encoded indices:

- the previous encoded index of the same source, in the same spectral band: $i_{m,n-1}^k$.
- the encoded index of the previous source, in the same spectral band: $i_{m-1,n}^k$.
- the encoded index of the same source, in the previous spectral band: $i_{m,n}^{k-1}$.

This dependence makes *optimal* CEC encoding infeasible since it needs to search over all possible $\prod_{k=1}^{K} \prod_{m=1}^{M} (L_m^k)^N$ image encodings. A practical greedy encoding algorithm is usually used to avoid the exhaustive encoding.

Note that in (3.1), $\underline{i_m^k}$ affects $D_m^k(\underline{i_m^k})$, $R_m^k(\underline{i_m^k}, \underline{i_{m-1}^k}, \underline{i_m^{k-1}})$, $R_{m+1}^k(\underline{i_{m+1}^k}, \underline{i_m^k}, \underline{i_{m+1}^{k-1}})$, and $R_m^{k+1}(\underline{i_m^{k+1}}, \underline{i_{m-1}^{k+1}}, \underline{i_m^k})$. Thus, instead of minimizing

$D_m^k(\underline{i_m^k}) + \lambda R_m^k(\underline{i_m^k}, \underline{i_{m-1}^{k,(0)}}, \underline{i_m^{k-1,(0)}})$, which is performed in a greedy encoding technique, we minimize $D_m^k(\underline{i_m^k}) + \lambda R_m^k(\underline{i_m^k}, \underline{i_{m-1}^{k,(0)}}, \underline{i_m^{k-1,(0)}}) + \lambda R_{m+1}^k(\underline{i_{m+1}^{k,(0)}}, \underline{i_m^k}, \underline{i_{m+1}^{k-1,(0)}}) + \lambda R_m^{k+1}(\underline{i_m^{k+1,(0)}}, \underline{i_{m-1}^{k+1,(0)}}, \underline{i_m^k})$ instead. Note that the superscript (0) denotes the solution obtained at time 0. This can be done via dynamic programming.

Because of the DPCM loop, the sample sequences that are encoded by the encoder are the error image, obtained by subtracting the original image by the predicted image. Since the encoded sequences for the current band are used to predict the next band, they produce the residual that will be encoded (quantized) for the next band. Effectively, the Lagrangian cost function changes every time the new predicted image is calculated. This makes the hillclimbing property of the algorithm invalid.

Iterative CEC Encoding for Hyperspectral Images

1. For the first band, $k = 1$, using greedy algorithm to obtain,

   $\underline{i_m^{1,(0)}} \quad \forall m.$

2. For each k = 2 to K, compute the error image.

   Using greedy algorithm to obtain, $\underline{i_m^{k,(0)}} \quad \forall m$. Set $t = 0$.

3. Do {

   - $t \leftarrow t + 1$
   - For the first band, $k = 1$, using dynamic programming,

     choose $\underline{i_m^{1,(t)}}$ that minimizes

     $D_m^1(\underline{i_m^{1,(t)}}) + \lambda R_m^1(\underline{i_m^{1,(t)}}, \underline{i_{m-1}^{1,(t)}})$

     $+ \lambda R_{m+1}^1(\underline{i_{m+1}^{1,(t-1)}}, \underline{i_m^{1,(t)}}) + \lambda R_m^2(\underline{i_m^{2,(t-1)}}, \underline{i_{m-1}^{2,(t)}}, \underline{i_m^{1,(t)}})$

   - For k=2 to K-1

     - compute the error image.
     - Using dynamic programming, choose $\underline{i_m^{k,(t)}}$ that minimizes

       $D_m^k(\underline{i_m^{k,(t)}}) + \lambda R_m^k(\underline{i_m^{k,(t)}}, \underline{i_{m-1}^{k,(t)}}, \underline{i_m^{k-1,(t)}})$

       $+\lambda R_{m+1}^k(\underline{i_{m+1}^{k,(t-1)}}, \underline{i_m^{k,(t)}}, \underline{i_{m+1}^{k-1,(t)}}) + \lambda R_m^{k+1}(\underline{i_m^{k+1,(t-1)}}, \underline{i_{m-1}^{k+1,(t)}}, \underline{i_m^{k,(t)}})$

     End

   - For the last band, $k = K$, compute the error image, then using

     dynamic programming, choose $\underline{i_m^{K,(t)}}$ that minimizes

     $D_m^K(\underline{i_m^{K,(t)}}) + \lambda R_m^K(\underline{i_m^{K,(t)}}, \underline{i_{m-1}^{K,(t)}}, \underline{i_m^{K-1,(t)}})$

     $+ \lambda R_{m+1}^K(\underline{i_{m+1}^{K,(t-1)}}, \underline{i_m^{K,(t)}}, \underline{i_{m+1}^{K-1,(t)}})$

   } Until a convergence criterion is met

Note that the convergence of the above algorithm is not guaranteed because of the DPCM loop. In practice, the encoding is repeated until there is no major change is the Lagrangian cost function.

3.4 Experiment

This section explains how the experiment and simulations were set up. Also, it describes how the results, which will be discussed in the next section, are obtained.

A system in Figure 3.2 was used in the simulation. We considered three different encoder/decoder pairs: ECTCQ, CECTCQ, and iterative-CECTCQ.

To obtain an error image, the optimum first-order linear predictor is used to get a predicted image, which will be subtracted from the original image. The optimum first-order linear predictor has the following equation:

$$\tilde{x} = \rho x + \mu(1 - \rho) \tag{3.2}$$

Here, μ and ρ are the mean and correlation coefficient of the input sequence, respectively. Based on [2], data obtained by AVIRIS have the spectral correlation coefficient, for any given spatial pixel, approximately 0.95. Thus, this value is used in the predictor.

For each spectral band, the 2-D DCT is applied to each of 8×8 pixels blocks of the error image. The transformed coefficients at each spatial location are collected to obtain source sequences. For a 256×256 AVIRIS image, with 8×8 pixels transform blocks, there are 64 source sequences, each with a length of 1024 samples, for each spectral band.

For codebook design and rate allocation, the transformed coefficient sequences are assumed to have various generalized Gaussian statistics [2]. The codebooks were designed

by using generalized Gaussian pseudo random numbers as training data. Generalized Gaussian random numbers have the following probability density function [2]:

$$f_x(x) = \left[\frac{\alpha\eta(\alpha,\sigma)}{2\Gamma(1/\alpha)}\right] \exp\{-[\eta(\alpha,\eta)|x|]^\alpha\} \tag{3.3}$$

where,

$$\eta(\alpha,\sigma) \equiv \sigma^{-1}\left[\frac{\Gamma(3/\alpha)}{\Gamma(1/\alpha)}\right]^{1/2}. \tag{3.4}$$

The shape parameter α describes the rate of exponential decay and σ is the standard deviation of the associated random variable [19]. The gamma function $\Gamma(\cdot)$ is defined as

$$\Gamma(n) = \int_0^\infty e^{-x}x^{n-1}dx. \tag{3.5}$$

[2] shows that excellent overall performance can be obtained by using codebooks optimized for generalized Gaussian distributions with $\alpha = 2.0$, and $\alpha = 0.75$ for the DC and non-DC coefficient sequences, respectively. These distributions are used to design codebooks by using a modified version of the generalized Lloyd algorithm to minimize the Lagrangian cost function (equation 3.1).

Test data from AVIRIS [3] are used in this chapter in order to make a reasonable comparison with other methods. The data are lowpass filtered and resized to 256×256 pixels to reduce computational complexity. We used only the first 100 bands of the data. The distortion and the rate were measured and recorded at the encoder for each band. The results are plotted in the next section.

[3] http://makalu.jpl.nasa.gov/aviris.html.

3.5 Results

In this section, a discussion of results based on the experiments explained in section 3.4 is given.

Figure 3.4 shows coding results, in terms of peak signal to noise ratio (PSNR), for different encoding methods, at 0.1 bits per pixel per band, for a 256×256 AVIRIS image. The PSNR for each encoding scheme is plotted versus different spectral bands. The PSNR (in dB) is calculated by $10 * \log_{10} \frac{255^2}{E\{(s_n - q(i_n))^2\}}$. We compared the results produced by 1) ECTCQ, 2) greedy CECTCQ, and 3) iterative-CECTCQ. For the iterative-CECTCQ, the result is obtained after convergence, which is usually after 3-4 iterations. We say the algorithm converge when the percentage differential in the cost function is less than 10^{-6}. We choose to show a range of the 30^{th} band to the 70^{th} band for comparison with the results from [2] and [1]. It can be clearly seen that each band has different encoding performance. The average PSNR for all 100 bands is 40.2 dB, 40.76 dB and 41.64 dB, for ECTCQ, CECTCQ, and iterative-CECTCQ, respectively. Note that the performance from the 54^{th} band to the 57^{th} band is worse than other bands. This is due to the fact that the particular image bands have a high sensor noise which can be seen from the image. This fact is also observed in [2] and [1].

Generally, the encoding performance of the greedy CECTCQ is always better than that of ECTCQ, for all bands. The gain is almost 1dB over ECTCQ at some spectral bands. Since a third-order probability model is used in CECTCQ, the required memory is higher, compared to ECTCQ.

The iterative CECTCQ gives the best encoding performance for all bands. A gain of almost 2dB over ECTCQ can be observed in many spectral bands. The performance gain obtained from the iterative CECTCQ, compared to the greedy CECTCQ, comes at a cost of increasing computational complexity. Running a non-optimized C code on Space machine takes 352 seconds for the greedy CECTCQ, while it takes 3992 seconds for the iterative CECTCQ (4 iterations). The complexity is roughly '2 $\times$ number of iterations + 1' times greater than the greedy CECTCQ.

3.6 Conclusion

In this chapter, the iterative hillclimbing image encoding algorithm proposed in chapter 2 was extended to encode hyperspectral images. Since hyperspectral images can be thought as a stack of highly correlated images of different spectral bands, the encoder can capture redundancies within the band (similar to the previous chapter) and redundancy between spectral bands. By using a third-order probability model to capture all dependencies, including the dependencies between frequency bands, the iterative hillclimbing encoder was naturally extended to encode hyperspectral images. Since the hybrid DPCM/DCT system was used, the hillclimbing property of the algorithm introduced in chapter 2 is no longer valid. Nevertheless, the result shows an improvement over the greedy algorithm.

For future research direction, a DPCM with linearly filtered past values of the quantized error [57] or an adaptive DPCM system might be used. An extension to this chapter is to encode *column-wise*, similar to the previous chapter. One can even try to encode *z-axis wise*.

The next chapter will demonstrate that the same iterative principle can be applied to a noisy image decoding application. The cost function that needs to be optimized will be different, but the iterative optimization principle remains the same.

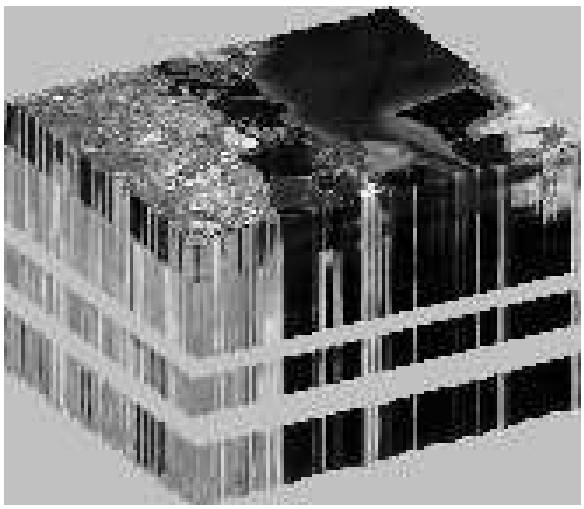

Fig. 3.1. A hyperspectral image cube. Each horizontal "slice" represents the same spatial area.

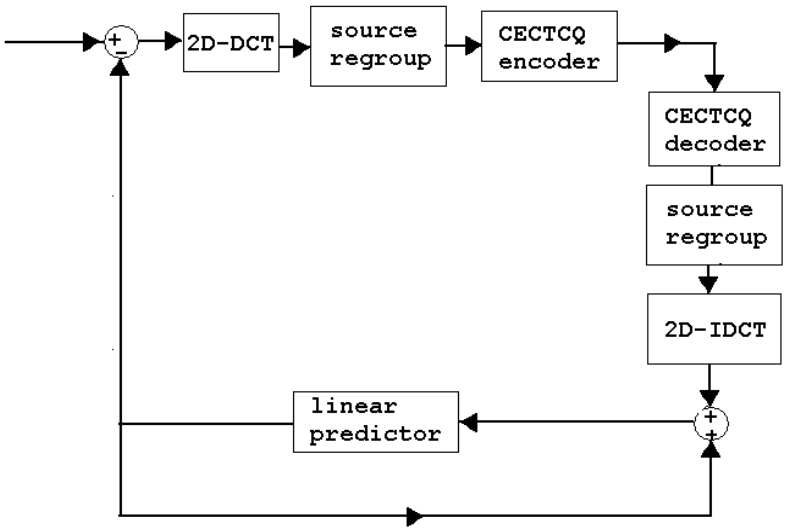

Fig. 3.2. A hybrid DPCM/DCT system for hyperspectal image encoder/decoder.

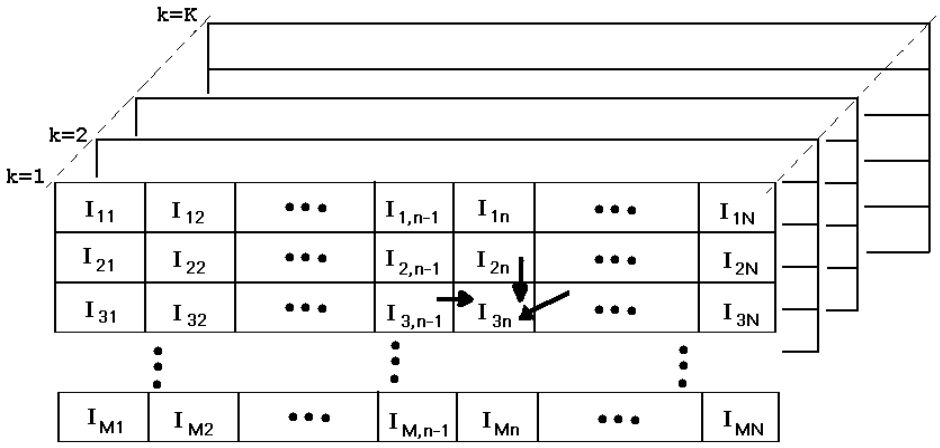

Fig. 3.3. 3-Dimension of source sequences, with a total of K bands. Arrows indicate dependencies for one particular sample. For this chapter, the source sequences are the error sequences from the DPCM loop.

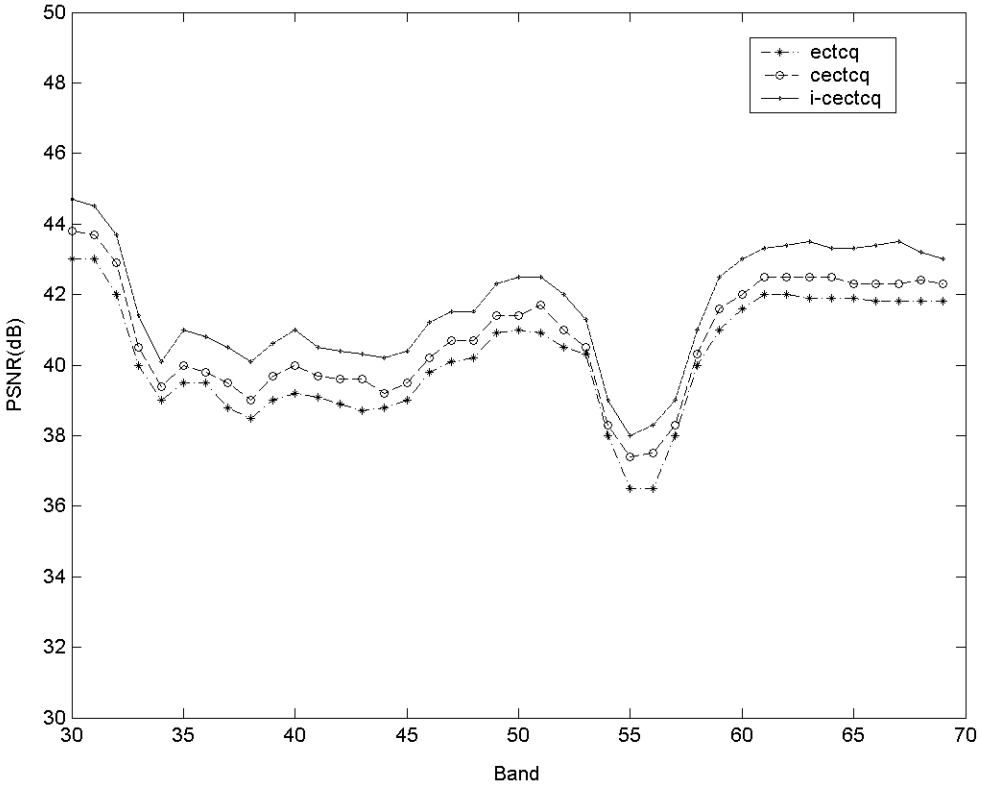

Fig. 3.4. PSNR vs Spectral Bands results on 256 × 256 AVIRIS image, at 0.1 bit per pixel per band.

Chapter 4

Iterative Hillclimbing Noisy Channel Decoder For Transform Images

4.1 Introduction

In previous chapters, a new iterative optimization algorithm was proposed to iteratively optimize transformed image encoders. In this chapter, we will demonstrate that, with slight modification, the very same optimization technique can be effectively used in noisy image decoding applications.

In most communication systems, encoded indices are passed to a decoder via a noisy environment. The objective of a decoder is to estimate the transmitted symbols (indices) from the received symbols, which are likely to be corrupted with noise. A simple (and suboptimal) decoder assumes received indices are correct (noiseless), then uses them for lookup table decoding. The complexity of this decoder is very minimal. It only requires a lookup table which is usually obtained via a different noiseless channel as side information. Obviously, the solution obtained by this kind of decoder is suboptimum since the decoder ignores the effect of the channel. A better decoder uses sequence maximum *a posteriori* probability (MAP) decoding technique to estimate the optimal decoded sequence [56]. The decoder searches for the most likely sequence of transmitted indices by using a dynamic programming algorithm. The solution obtained from this kind of decoder is optimum in the sense of a sequence likelihood function. However, the

computational complexity of this decoder is very high since it needs to keep track of the accumulated cost for every state.

In the case of transformed image transmission, each transform coefficient source sequence can be independently transmitted, i.e. source by source, through a noisy channel. As discussed in the previous chapter, there are redundancies (dependencies) between coefficient sources that a decoder can take advantage of. At the decoder side, if a simple (naive) decoder is implemented, each transmitted symbol will be decoded one by one, without having to wait to receive the whole source sequence. However, this simple decoder does not consider dependencies between sources. To find the optimal decoded solution taking account of known spatial dependencies, one has to do an exhaustive search over all possible solutions, just as in the case of image encoding. Clearly, the computational complexity is intractable. It will be shown later that the computational complexity grows exponentially with the number of coefficient sources.

Another possibility of decoding a transformed image is to use sequence MAP decoders to decode each source sequence independently (i.e. ignore the source dependencies). A sequence MAP decoder searches for a solution for *each* coefficient sequence, one sequence at a time. To further improve the decoding performance, without increasing too much the computational complexity, one can use a sequence MAP decoder to search for a solution one source at a time, using a conditional probability model to capture the source dependencies. This is called a conditional sequence MAP decoder. Since the decoder finds a solution for one source at a time, the overall computational complexity grows linearly with the number of sources. The obtained solution is optimum for each source, but not optimum for the overall image. Similar to chapter 2, this decoding

technique does not guarantee even a locally optimal solution. This type of decoder will be a reference model for this chapter.

The iterative optimization method introduced in chapter 2 for encoding transform images can be modified for noisy channel decoding of transformed images as well. After applying a conditional sequence MAP decoding for all source sequences, it is possible to go back and decode them again, using the new information from the previous iteration. The iteration allows the decoder to readjust the decoded samples to improve the likelihood function. The next section will describe how to modify our iterative hillclimbing algorithm for decoding transformed images. Sections 4.3 and 4.4 show how to set up the experiment and explain the results, respectively. Lastly, conclusions are given in section 4.5.

4.2 Formulation

This section gives a mathematical model of the reference algorithms and the proposed algorithm.

Let $\{\underline{z} = \underline{z_1}, \underline{z_2}, \ldots, \underline{z_M}\}$, where $\underline{z_m} \equiv (z_{m1}, z_{m2}, \ldots, z_{mN})$, $z_{mn} \in \{1, \ldots, L_m\}$, ($L_m$ is the number of quantization levels for source m), be the received sequences for all transform coefficient sources. Also, let $\{\underline{\mathbf{i}} = \underline{i_1}, \underline{i_2}, \ldots, \underline{i_M}\}$, where $\underline{i_m} \equiv (i_{m1}, i_{m2}, \ldots, i_{mN})$, $i_{mn} \in \{1, \ldots, L_m\}$, be the transmitted sequences for all sources. The decoder tries to estimate $\{\underline{i_1}, \underline{i_2}, \ldots, \underline{i_M}\}$, denoted by $\{\underline{\hat{i}_1}, \underline{\hat{i}_2}, \ldots, \underline{\hat{i}_M}\}$.

In the transformed image case, after the noisy channel, received sample sequences, denoted by $\underline{Z}$, can be displayed as a two-dimensional array as in Figure 4.2. Note that this is similar to the array in chapter 2. The difference is the samples now are received

indices, instead of encoded indices. The redundancy between sources still exists, however. For an optimum MAP decoder, the decoder finds the optimal solution which maximizes $P[\underline{I}|\underline{Z}]$. The decoder needs to search over $(\prod_{m=1}^{M}(L_m)^N)$ solutions, where M is the number of sources, and N the number of samples in each source. Obviously, it is impractical.

If source-by-source (row-by-row) decoding is performed, and independence between source sequences is assumed, a sequence MAP decoder finds the optimal solution for source m by optimizing $P[\underline{I_m}|\underline{Z_m}]$. Note that this decoding method does not utilize dependencies between sources.

In a conventional JSC decoder, the joint likelihood function for the mth source is:

$$\begin{aligned} L_m &= P[\underline{z_m}, \underline{i_m}] \\ &= P[z_{m1}|i_{m1}]P[i_{m1}] \prod_{n=2}^{N} P[z_{mn}|i_{mn}]P[i_{mn}|i_{m,n-1}] \end{aligned} \tag{4.1}$$

We can see that $\{P[i_{mn}|i_{m,n-1}]\}$ shows the dependency *within* the source. Since an independence between sources is assumed, the total likelihood function can be expressed as a product of L_m for all m. Or in another form, the log-likelihood function can be express as:

$$\begin{aligned} LL &= \log\{\prod_{m=1}^{M} P[\underline{z_m}, \underline{i_m}]\} \\ &= \sum_{m=1}^{M} \log\{P[\underline{z_m}, \underline{i_m}]\} \end{aligned} \tag{4.2}$$

To utilize the dependency between sources, a conditional probability model is used in dynamic programming. For simplicity, a second-order Markov probability model is

assumed, i.e. the model will be based on the probabilities $\{Prob[i_{mn}|i_{m,n-1}, i_{m-1,n}]\}$. This leads to a new likelihood function:

$$\begin{aligned} L'_m &= P[\underline{z_m}, \underline{i_m}|\underline{i_{m-1}}] \\ &= P[z_{m1}|i_{m1}]P[i_{m1}|i_{m-1,1}] \prod_{n=2}^{N} P[z_{mn}|i_{mn}]P[i_{mn}|i_{m,n-1}, i_{m-1,n}] \quad (4.3) \end{aligned}$$

Note that the conditional probabilities are used to represent redundancy between sources m and $m-1$, as well as within each source. The overall log-likelihood function can be expressed as:

$$LL' = \log\{P[\underline{z_1}, \underline{i_1}]\} + \sum_{m=2}^{M} \log\{P[\underline{z_m}, \underline{i_m}|\underline{i_{m-1}}]\}. \quad (4.4)$$

For a greedy JSC decoder, it first decodes the first received source sequence ($m = 1$) by searching for optimum sequence in term of likelihood function by using dynamic programming to optimize (4.1). For the subsequent sources, the greedy decoder decodes one at a time, by using dynamic programming to optimize (4.3).

For the same reason as in previous chapters, this greedy algorithm is not optimal, even though it finds the optimum sequence for *each* source sequence individually, given past choices. It is possible to revisit decoded sequences and adjust to improve the overall cost function (likelihood function). However, if the decoder re-decodes the same source sequences with the same log-likelihood function, the same decoded sequences will be obtained. However, for source m, the decoder can use dynamic programming to find $\underline{i_m}$ to optimize $\log\{P[\underline{z_m}, \underline{i_m}|\underline{i^{(0)}_{m-1}}]\} + \log\{P[\underline{z_{m+1}}, \underline{i^{(0)}_{m+1}}|\underline{i_m}]\}$. For the same reason stated in chapter 2, this guarantees that $\underline{i^{(1)}_m}$ must be a least as good as $\underline{i^{(0)}_m}$ in the log-likelihood sense. Thus, we propose an iterative hillclimbing JSC decoding technique.

Iterative, hillclimbing JSC Decoding

1. Implement the normal decoding procedure to maximize L_m

   to determine $\{\underline{\hat{i}_m^{(0)}}, m = 1, \ldots, M\}$.

   Set $t = 0$.

2. Do {

   - $t \leftarrow t + 1$
   - Using dynamic programming to maximize

     $\log\{P[\underline{z_1}, \underline{i_1^{(t)}}]\} + \log\{P[\underline{z_2}, \underline{i_2^{(t-1)}} | \underline{i_1^{(t)}}]\}$ to get $\underline{\hat{i}_1^{(t)}}$.

   - For m=2 to M-1

     – Using dynamic programming to maximize

       $\log\{P[\underline{z_m}, \underline{i_m^{(t)}} | \underline{i_{m-1}^{(t)}}]\} + \log\{P[\underline{z_{m+1}}, \underline{i_{m+1}^{(t-1)}} | \underline{i_m^{(t)}}]\}$

       to get $\underline{\hat{i}_m^{(t)}}$.

     End

   - Using dynamic programming to maximize

     $\log\{P[\underline{z_M}, \underline{i_M^{(t)}} | \underline{i_{M-1}^{(t)}}]\}$ to get $\underline{\hat{i}_M^{(t)}}$.

   }

3. Repeat Step 2 until convergence

4.3 Experiment

This section explains how the simulation is set up and how the results are obtained.

The system in Figure 4.1 is the reference system used in this simulation. We have tested our method for noisy transform image decoding, in comparison with a naive, a sequence MAP, and a greedy conditional sequence MAP decoder.

On the encoder side, we used 8×8 image blocks, the 2-D DCT transform, and a zig-zag scan for 1-D ordering of the sources. Four-state TCQ was used on each source, with the trellis as given in the original TCQ paper [44]. The codebook size for each source was determined by a bit allocation strategy. A training set of 8 images (similar to the ones in chapter 2) was used for designing the coders (the TCQ codebooks), and estimating probabilities. We designed the encoder to operate at a rate of 1 bit per sample.

A binary symmetric channel (BSC), with different bit error rates, was implemented as a noisy channel.

On the decoder side, different decoding techniques were implemented. For a naive decoder, the received sequences were assumed to be correct. Using a quantizer lookup table, the decoded sequence was compared with the source sequence and the peak-signal-to-noise-ration (PSNR) was computed.

For a sequence MAP decoder, after receiving a complete sequence for each source, a dynamic algorithm was applied to maximize (4.1), for each m. Again, a quantizer lookup table was used and the PSNR was computed from the decoded sequence.

For a greedy conditional sequence MAP decoder, probabilities $\{Prob[i_{mn}|i_{m,n-1}, i_{m-1,n}]\}$ are used. These probabilities are estimated by the training images. The first received sequence was decoded by a dynamic programming which maximizes (4.1). All the remaining source sequences were decoded by using a

dynamic programming which maximizes (4.3). Table lookup was then used. The PSNR was then computed.

Finally, for our iterative JSC decoding, the (just described) standard greedy conditional sequence MAP decoding technique was used to obtain initial decoded sequences. Then, the proposed iterative decoding algorithm was used to iteratively decode each source at a time. The decoding algorithm was repeated until either there are no further changes or until a stopping condition is reached. This iterative decoding algorithm is non-decreasing in the likelihood function.

The resulting curves are obtained by implementing the various decoders for a sequence of channel bit error rate values. These curves will be plotted and discussed in the result section.

4.4 Results

In this section, a discussion is given on the results obtained by performing the experiment described in section 4.3.

Figure 4.3 and 4.4 show the plot of a peak signal-to-noise ratios (PSNR) versus a range of channel bit error rate for the training data set and test data set (Lena image), respectively. The peak signal-to-noise ratio is calculated by using $PSNR(dB) = 10\log_{10}(\frac{255^2}{mse})$. The plot compares the PSNRs for a naive, a sequence MAP, a conditional sequence MAP, and an iterative sequence MAP decoder. For the iterative sequence MAP decoder, the final iteration result is plotted. From the experiment, the result does not show an improvement after 3-4 iterations.

We tested the system with the channel bit error rate ranges from 10^{-3} to 3×10^{-1}.

As expected, the higher the channel noise level (high BER), the lower the performance of all decoders (lower PSNR). At the channel bit error rate lower than 4×10^{-3}, all decoders have the same performance on the training data. For the test data, all decoders perform equally for the channel bit error rate lower than 2×10^{-2}. For both training set and test set, the naive decoder always performs the worst. The iterative sequence-MAP decoder always performs equally or better than other decoders. At the channel BER of 2×10^{-1}, the iterative sequence-MAP decoder outperforms the naive decoder by as much as 2 dB. The higher the noise level, the more gain we obtain from the iterative sequence-MAP decoder, comparing to the naive decoder.

In the test set, the sequence-MAP decoder outperforms the naive decoder by more than 1.5 dB at the channel BER of 2×10^{-1}. At the same channel BER, the iterative sequence-MAP decoder outperforms the sequence-MAP decoder by 0.5 dB.

We noticed that most gain is at high noise levels, which means the decoded image quality may be useless, regardless of the method. One explanation for this minimal gain is that we optimize the likelihood function, which is not equal to optimzing the mean-squared error. One might try to use an MMSE decoder to improve the decoding performance.

4.5 Conclusion

This chapter illustrates that the iterative optimization technique that was introduced in chapter 2 for image encoders, and again in chapter 3 for hyperspectral image encoders, can be effectively applied to noisy channel image decoder applications.

Since the proposed decoder is an iterative decoder, all sequences (rows) are needed for decoding. Thus, there is a high decoding delay for this method. This is different than a sequence MAP decoder, where it only needs one sequence at a time.

In general, encoders try to find the best representation of input samples, given a constraint in distortion and rate. Decoders try to find the most likely transmitted symbols from noisy received symbols. While there is a big difference between encoders and decoders, the proposed optimization technique can be applied in both applications. This is because the underlying idea of the iterative optimization is to readjust the previous decision (encoding or decoding) to improve the cost function.

To further explore this topic, one can try to decode *column-wise* which is very similar to what we did in chapter 2.

Fig. 4.1. A communication system consists of a source encoder, a noisy channel and a source decoder.

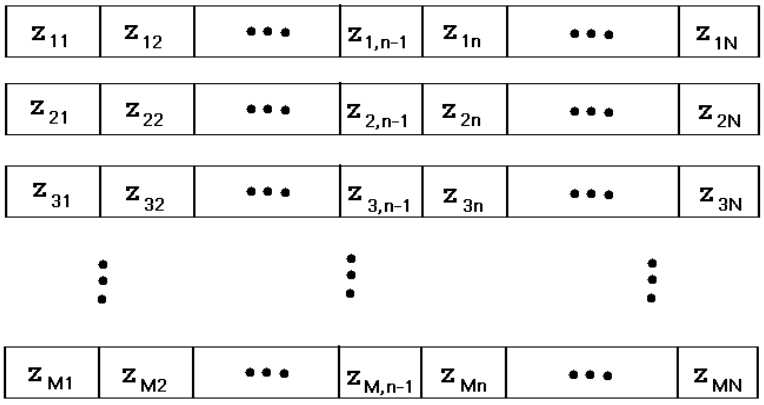

Fig. 4.2. 2D representation of received source sequences.

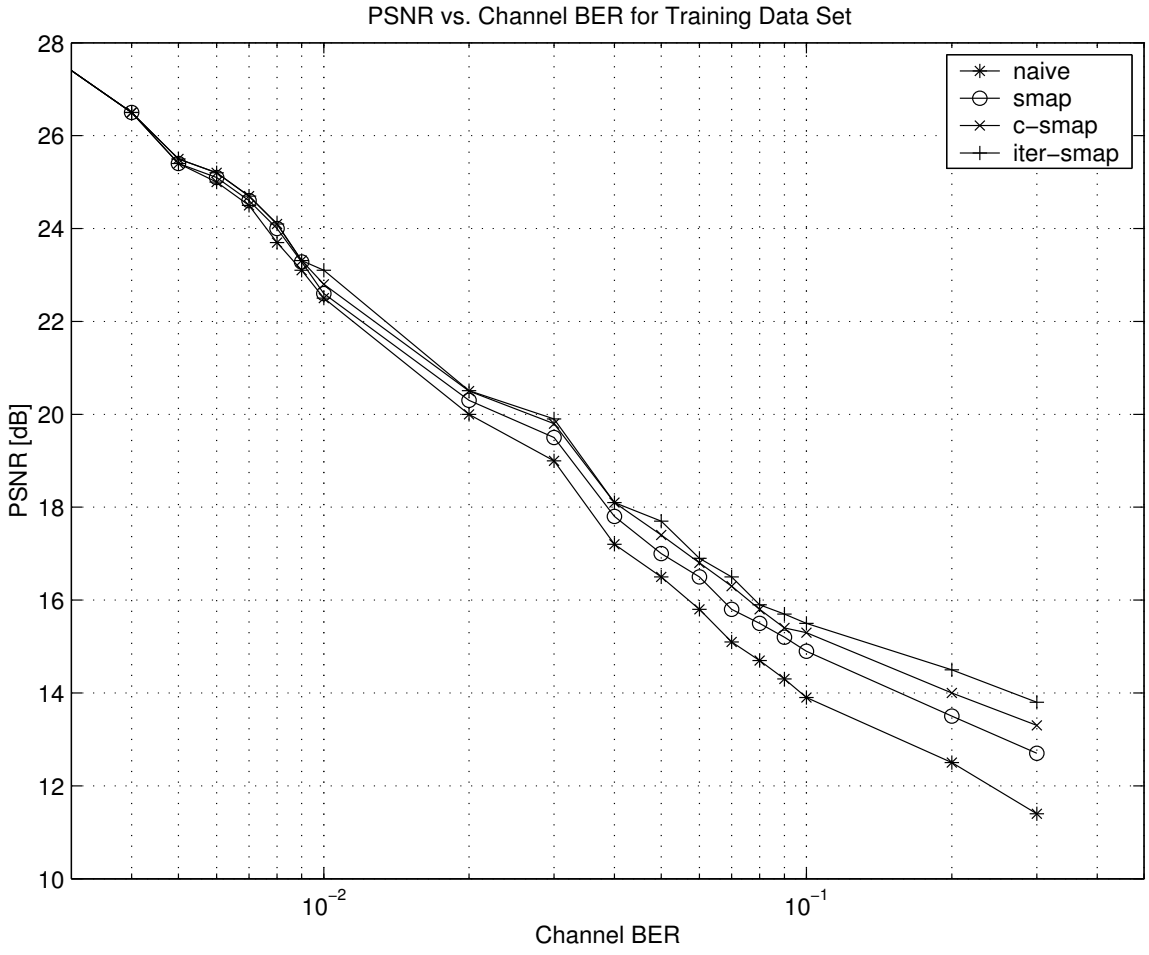

Fig. 4.3. The result for the training data set. PSNR vs channel bit error rate results for a naive, a sequence MAP, a conditional sequence MAP and an iterative JSC decoder.

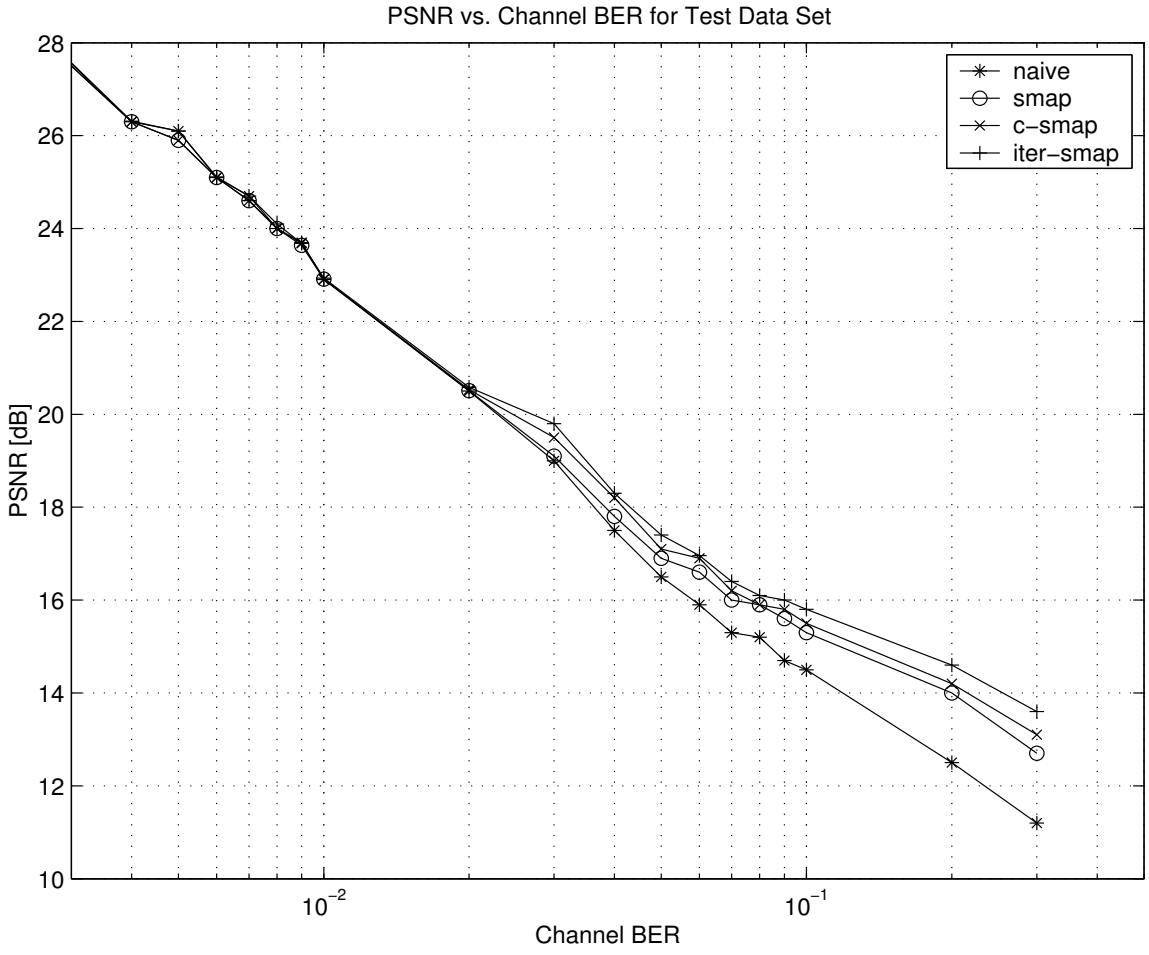

Fig. 4.4. The result for the testing data set. PSNR vs channel bit error rate results for a naive, a sequence MAP, a conditional sequence MAP and an iterative JSC decoder.

Chapter 5

Turbo-type Joint Source-Channel Decoding

5.1 Introduction

In chapter 4, an iterative-type image decoder was proposed for image decoding over a noisy channel. A simple communication system consisting of a source encoder, a discrete noisy channel, and a source decoder was studied. The decoding performance was improved by up to 2 dB over a standard greedy decoding technique. However, in most communication systems, like the one shown in Figure 5.1, a channel encoder is usually added to increase redundancy of encoded symbols to combat channel noise. On the decoder side, a channel decoder is implemented to decode the corrupted bit stream. The output of the channel decoder is much *cleaner*; thus a source decoder usually performs much better when a system uses channel coding than if it does not. Due to the fact that the channel encoder and decoder both work with bit information, we call them bit-level encoders/decoders. Likewise, the source encoder and decoder are described as working at the symbol-level.

Based on the system in Figure 5.2, a new kind of iterative decoder is introduced. The basic idea of this decoder comes from turbo-type iterative decoding techniques. Specifically, the parallel concatenation structure of a turbo-code decoder introduced in [8] is modified so that the first stage is a bit-level decoder and the second stage is a symbol-level decoder (Figure 5.2).

A similar idea was also implemented in [7]. The authors studied iterative joint source/channel decoding applied to a variable length coded text source. A text consisting of 82 characters out of the ASCII set was encoded by a symmetrical reversible variable length code (RVLC) encoder to generate variable length codewords (VLCs). These codewords were mapped to binary vectors which were then permuted in the interleaver before entering the channel encoder. A VLC-trellis was introduced by the authors and utilized by the decoder. The channel-encoded vector was transmitted over an additive-white-Gaussian-noise channel.

At the receiver side, the received (noisy) vector was decoded by a symbol-by-symbol *a posteriori* probability (APP) decoder using the BCJR algorithm. The output was deinterleaved and passed to the outer decoder which is also a BCJR decoder based on the "bit-level trellis". The output of the outer decoder produced APP values for every bit in the variable length vector. The calculated APP can be fed back, with proper interleaving, to the inner decoder to improve the performance. The result shows that at an error rate of 8×10^{-5}, a gain of more than 2dB is obtained over a basic scheme, with only 4 iterations.

In this chapter, a model similar to the one in [7] will be introduced and studied. The differences between these two models are:

- A fixed-length code is studied in this chapter; thus there is no VLC-trellis. Instead, there is an FLC trellis. However, the material in this chapter can be extended for the VLC case as well.

- A sequence-based approximate minimum mean square estimator (SAMMSE) [54] is used at the outer decoder to produce APPs for each *symbol*, while the outer decoder in [7] produces APPs for each *bit*.

- A synthesized first-order Gauss-Markov source is used in this chapter.

Despite the common term *iterative* used in this chapter and in chapter 4, there is a big difference in the underlying idea. In chapter 4, the iteration is in the source (symbol) decoder. The decoder iterates over the whole received streams (rows), revisiting the previously decoded symbols to improve the overall cost function. Dependencies between sources are modelled by conditional probabilities and are included in the cost function.

In this chapter, the term iterative is used to describe the interaction between bit-level and symbol-level decoders. The decoder iteratively decodes the whole received sequence with a new update of APP from each decoder stage to improve the likelihood function. Although, mathematically, there is no proof that the likelihood function will be improved with the iteration, the experimental results show a noticeable improvement after few iterations.

Note that in this chapter, we do not exploit the dependency between source sequences (e.g. neighboring rows of an image). In particular, we only consider a 1-D source sequence.

The next section reviews the Turbo-type decoding technique. Then we will describe the formulation and algorithm for this new iterative decoder in detail. Experiments and simulations are explained in section 5.3. Results and conclusions are in sections 5.4 and 5.5, respectively.

5.2 Turbo-type Joint Source-channel Decoder

In this section, a new iterative JSC decoding technique based on turbo decoding is introduced. The basic idea and formulation are given in this section. It is natural to give a review of decoding techniques of turbo codes first.

5.2.1 Review of Turbo Decoding

Turbo-codes were first introduced in [8] as concatenated convolutional codes to increase coding gain. It is constructed by parallel combining of two recursive systematic convolutional (RSC) codes. Figure 5.3 shows a turbo encoder with a rate of 1/3. The RSC encoders are not necessarily identical. The data bit at time k ,b_k, directly enters the first RSC encoder. It passes through an interleaver, to get data bit at time n, b_n, before entering the second RSC encoder. Data b_k is systematically transmitted as symbol D_k and *redundancies* DY_{1k} and DY_{2k} are produced by the first and second RSC encoders, respectively.

There are two types of decoding techniques: a serial concatenation scheme, and a parallel concatenation scheme (see Figure 5.4). Both have two stages, one trying to find likelihood ratio for b_k while the other for b_n. Thus, the output of each stage is a log-likelihood ratio,

$$\bigwedge(b_k) = \log \frac{\text{Prob}[\text{b}_\text{k} = 1/\text{received sequence}]}{\text{Prob}[\text{b}_\text{k} = 0/\text{received sequence}]}, \tag{5.1}$$

where $\text{Prob}[\text{b}_\text{k} = \text{j/received}], \text{j} = 0, 1$ is the APP of bit b_k.

From (5.1), the APP of each bit is needed. A modified Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm for RSC codes [8] is thus used in each decoder stage to obtain such probabilities.

5.2.1.1 Modified BCJR for RSC Codes:

This section reviews the algorithm given in [8].

Consider a discrete time memoryless Gaussian noisy channel. The output of the channel, which is the input of the decoder, is denoted by $R_1^N = \{R_1, \ldots, R_k, \ldots, R_N\}$, with N the total number of bits. Thus, at bit time k, the input of the decoder is $R_k = (x_k, y_k)$, where

$$\begin{aligned} x_k &= (2D_k - 1) + n_{1k} \\ y_k &= (2DY_k - 1) + n_{2k}. \end{aligned} \tag{5.2}$$

n_{1k} and n_{2k} are two independent noise with the same variance σ^2, and DY_k is $DY1_k$ or $DY2_k$, depending on the decoder stage. Furthermore, consider a RSC code with constraint length K. Let $S_k = m$ represent the encoder is in the state m at bit time k.

The log-likelihood ratio can be calculated by using the following three parameters: forward probabilities, α, backward probabilities, β, and state transition probabilities,γ. Both α and β are expressed in terms of γ; thus we need to first calculate γ as follows:

$$\begin{aligned} \gamma_j(R_k, m', m) &= p(R_k | b_k = j, S_k = m, S_{k-1} = m') \\ &\quad \cdot q(b_k = j | S_k = m, S_{k-1} = m') \\ &\quad \cdot \pi(S_k = m | S_{k-1} = m'), \end{aligned} \tag{5.3}$$

where the first term is the channel output probability of the discrete Gaussian memoryless channel. The second term is equal to 0 or 1 since the convolutional encoder is a deterministic machine. If the encoder inputs are equiprobable, the last term will be 1/2 for each of these transitions. Note that some state transitions are invalid, depending on the encoder structure.

Both α and β can be recursively expressed in terms of γ as follows [8]:

$$\alpha_k(m) = \frac{\sum_{m'} \sum_{j=0}^{1} \gamma_j(R_k, m', m)\alpha_{k-1}(m')}{\sum_m \sum_{m'} \sum_{j=0}^{1} \gamma_j(R_k, m', m)\alpha_{k-1}(m')} \tag{5.4}$$

$$\beta_k(m) = \frac{\sum_{m'} \sum_{j=0}^{1} \gamma_j(R_{k+1}, m', m)\beta_{k+1}(m')}{\sum_m \sum_{m'} \sum_{j=0}^{1} \gamma_j(R_{k+1}, m, m')\alpha_k(m')} \tag{5.5}$$

Using the above three parameters, the log-likelihood ratio can be calculated by

$$\bigwedge(b_k) = \log \frac{\sum_m \sum_{m'} \gamma_1(R_k, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_0(R_k, m', m)\alpha_{k-1}(m')\beta_k(m)}. \tag{5.6}$$

Modified BCJR for RSC Codes:

- Step 0: Initialization.

$$\begin{aligned} \alpha_0(0) &= 1 \\ \alpha_0(m) &= 0 \quad \forall m \neq 0 \\ \beta_N(m) &= \frac{1}{N} \quad \forall m. \end{aligned}$$

- Step 1: Forward probability calculation.

  For each R_k, $\gamma_j(R_k, m', m)$ and $\alpha_k(m)$ are computed by (5.3) and (5.4), respectively.

- Step 2: Backward probability calculation.

  When R_1^N has been completely received, $\beta_k(m)$ is computed using (5.5), and the log-likelihood ratio associated with each decoded bit b_k is computed by (5.6).

In [8], it was shown that , if the noise at the input bits are independent at each time k, the log likelihood ratio of each output bit is the sum of the log likelihood ratio of that bit at the decoder input and *extrinsic* information. Conditioning on b_k, both x_k and y_k are uncorrelated Gaussian variables. And because of the encoder is systematic, $\mathrm{Prob}[\mathrm{x_k}/\mathrm{b_k} = 1, \mathrm{S_k} = \mathrm{m}, \mathrm{S_{k-1}} = \mathrm{m}']$ is independent of S_k and S_{k-1}. Thus we can rewrite (5.6) as:

$$\bigwedge(b_k) = \log \frac{\mathrm{Prob}[\mathrm{x_k}/\mathrm{b_k} = 1]}{\mathrm{Prob}[\mathrm{x_k}/\mathrm{b_k} = 0]} + \log \frac{\sum_m \sum_{m'} \gamma_1(y_k, m', m)\alpha_{k-1}(m')\beta_k(m)}{\sum_m \sum_{m'} \gamma_0(y_k, m', m)\alpha_{k-1}(m')\beta_k(m)}. \qquad (5.7)$$

The second term in (5.7) is the extrinsic information, $W_k = \bigwedge(b_k)|_{x_k=0}$. This extrinsic information is introduced by residual redundancy of the encoder. In case of turbo-code, this extrinsic information is introduced by the parallel RSC encoder structure.

After applying the modified BCJR algorithm at the first stage, a hard decision is made by setting a threshold to the log-likelihood ratio. Note that the first stage decoder uses only partial information (i.e. D and DY_1). The second stage can also use the modified BCJR algorithm with D and a redundancy DY_2 as inputs to obtain a likelihood ratio of b_k. But the second decoder can perform better if it utilizes the decoded information, *assuming correct decoding*, from the first stage. Specifically, with proper interleaving, the decoded bit b_k from the first stage and DY_2 are used to find a log-likelihood ratio of b_n, using (5.6). For a serial concatenation scheme, a hard decision can

be made from the obtained log-likelihood ratio right after second stage (i.e. a threshold on a likelihood ratio). Alternately, for a better performance, an MMSE type decoder can be used to reconstruct the samples.

It can be noted that a serial scheme does not give the optimum solution, since the first stage uses only part of redundancy, i.e. DY_{1k}, to produce a likelihood function of b_k, while the second stage uses b_k and DY_{2k} as inputs. The decoding performance can be improved by using parallel concatenation scheme. For parallel concatenation scheme, the extrinsic information from the second stage is fed back ,with proper interleaving, to the first stage. Now, at the first stage, $R_k = (x_k, y_k, w_k)$, which can be thought as an extra redundancy that the first stage can utilize. The first stage repeats the decoding process. We can keep iterating until convergence. Although the solution obtained by the parallel scheme is not guaranteed to be optimal, [8] shows that the bit error rate decreases with each iteration. It also shows that the performance essentially converges after 4-5 iterations.

5.2.2 New Turbo-type Joint Source-channel Decoder

The last section shows how a parallel concatenation decoding scheme works. It is designed as a channel decoding technique. In this section, we propose a new turbo-type joint source-channel decoder. Basically, we modify the parallel decoding scheme for turbo-codes to perform joint source-channel decoding. The system in Figure 5.2 is the underlying system for this section. For a source encoder (symbol level encoder), we use a uniform scalar quantizer for simplicity. For a channel encoder (bit level encoder), a binary convolutional code is used. The main difference between the proposed decoder

and a parallel turbo code decoder is that the first stage of the proposed decoder is used as a channel decoder, and the second stage is used as a source decoder. Moreover, the main difference between the proposed system and the basic system in Figure 5.1, besides the feedback loop, is the soft information (i.e. probabilities) passed between decoder stages, instead of hard bit or symbol decisions.

Here is how the new turbo-type joint source-channel decoder works. From Figure 5.2, corrupted bit samples, $\underline{r}$, from noisy channel are *softly* decoded by the bit-level decoder (first stage). Instead of making hard bit decision, the bit APP's are calculated by the BCJR algorithm described in section 5.2.1.1.

The output of the first stage is a log-likelihood ratio. Similar to the parallel scheme in the standard turbo-code decoder, the log likelihood ratio of the output bit is the sum of the log likelihood ratio of that bit at the decoder input and the *extrinsic* information. Therefore, an extrinsic information must be extracted and used for the second stage decoder, which is a source (symbol) decoder. Note that, the APPs obtained from the first stage are at the bit level. However, symbol probabilities are needed at the symbol level decoder. By independence assumption, these *a posteriori* bit (extrinsic) probabilities are used to calculate symbol probabilities by

$$P[j_n|\underline{R}] = \prod_{l=1}^{length\ of\ j_n} P[b_{n,l}|\underline{R}], \tag{5.8}$$

where $b_{n,l}$ denotes the $l-th$ bit of $n-th$ symbol.

These symbol probabilities are then used in a *soft-decision* sequence-based approximate minimum mean-squared error (SAMMSE) decoder, introduced in [49], as a

symbol-level (second stage) decoder. The SAMMSE uses a forward-backward algorithm to find the *a posteriori* probabilities, $\mathrm{Prob}[\hat{\mathrm{i}}_\mathrm{n}|\underline{\mathrm{j}}]$, for each symbol time n.

5.2.2.1 Soft-decision SAMMSE

A sequence-based approximate MMSE decoder was introduced in [49] for source decoding over noisy channels. A sequence-based MMSE decoder first computes the *a posteriori* probability distribution for the set of source symbols; then the conditional mean estimator is applied to generate reconstruction values. In [49], it is argued that, given certain assumptions, this is the optimum decoder for the squared error cost. Because the reconstruction values are the average of the values in the codebook, they effectively use an infinite reconstruction "alphabet", unlike MAP detector-based decoders. Thus, the performance of the MMSE decoder is higher than that of the MAP decoder.

Next, we briefly review the formulation in [49]. Note that the mathematical background can be found in chapter 1. Consider a sequence of reproductions, $\underline{\hat{s}} = (\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_T), \hat{s}_t \in \mathcal{R}$, which the decoder uses to approximate the source sequence $\underline{s} = (s_1, s_2, \ldots, s_T)$. The distortion incurred by $\underline{\hat{s}}$ is $D(\underline{s}, \underline{\hat{s}}) = \sum_{t=1}^{T} d(s_t, \hat{s}_t)$. The decoder then selects the reconstruction levels according to the centroid rule [49]:

$$\hat{s}_t = \sum_{l=1}^{N} E[S_t|I_t = l]\mathrm{P}[\mathrm{I_t} = \mathrm{l}|\underline{\mathrm{J}} = \underline{\mathrm{j}}], \quad \forall \mathrm{t}. \tag{5.9}$$

Here,

$$P[I_t = l|\underline{J} = \underline{j}] = \frac{\sum_{\underline{i}:i_t=l} P[\underline{J} = \underline{j}|\underline{I} = \underline{i}]P[\underline{I} = \underline{i}]}{\sum_{\underline{i}} P[\underline{J} = \underline{j}|\underline{I} = \underline{i}]P[\underline{I} = \underline{i}]}. \tag{5.10}$$

Note that in (5.9), the probabilities $\mathrm{P}[\mathrm{I_t} = \mathrm{l}|\underline{\mathrm{J}} = \underline{\mathrm{j}}]$ are needed. A well-known foward/backward algorithm [5] can be used to calculate these probabilities. Forward

and backward probabilities can be recursively computed as follows:

$$
\begin{aligned}
\alpha_1[l] &= \mathrm{P}[\mathrm{I}_1 = \mathrm{l}]\mathrm{P}[\mathrm{J}_1 = \mathrm{j}_1|\mathrm{I}_1 = \mathrm{l}], \quad \mathrm{l} = 1, \ldots, \mathrm{N} \\
\alpha_t[l] &= \left[\sum_{k=1}^{N} \alpha_{t-1}[k]\mathrm{P}[\mathrm{I}_\mathrm{t} = \mathrm{l}|\mathrm{I}_{\mathrm{t}-1} = \mathrm{k}]\right] \cdot \mathrm{P}[\mathrm{J}_\mathrm{t} = \mathrm{j}_\mathrm{t}|\mathrm{I}_\mathrm{t} = \mathrm{l}], \\
&\qquad l = 1, \ldots, N, \quad t = 2, \ldots, T \\
\beta_T[l] &= 1, \quad l = 1, \ldots, N \\
\beta_t[l] &= \sum_{k=1}^{N} \mathrm{P}[\mathrm{I}_{\mathrm{t}+1} = \mathrm{k}|\mathrm{I}_\mathrm{t} = \mathrm{l}]\mathrm{P}[\mathrm{J}_{\mathrm{t}+1} = \mathrm{j}_{\mathrm{t}+1}|\mathrm{I}_{\mathrm{t}+1} = \mathrm{k}]\beta_{\mathrm{t}+1}[\mathrm{k}], \\
&\qquad l = 1, \ldots, N, \quad t = T-1, \ldots, 1.
\end{aligned} \tag{5.11}
$$

Finally, $\mathrm{P}[\mathrm{I}_\mathrm{t} = \mathrm{l}|\underline{\mathrm{J}} = \underline{\mathrm{j}}]$ can be rewritten in terms of α and β as:

$$
\mathrm{P}[\mathrm{I}_\mathrm{t} = \mathrm{l}|\underline{\mathrm{J}} = \underline{\mathrm{j}}] = \frac{\alpha_\mathrm{t}[\mathrm{l}]\beta_\mathrm{t}[\mathrm{l}]}{\sum_{\mathrm{m}=1}^{\mathrm{N}} \alpha_\mathrm{t}[\mathrm{m}]\beta_\mathrm{t}[\mathrm{m}]}. \tag{5.12}
$$

The above algorithm needs $\underline{J}$ to reconstruct the samples. [54] shows that the decoder performance can be further improved if soft information, i.e. *a posteriori* probabilities, are used at the input. In this case, after the first stage (bit-level) decoder, we can use $\mathrm{P}[\underline{\mathrm{J}}|\underline{\mathrm{R}}]$ to replace the channel transition probability in (5.6).

For the same reason as in the standard turbo-code decoder case, a feedback loop can be introduced to further improve the performance. To feed the extrinsic information back to the first stage decoder, a bit-level extrinsic information is first obtained from symbol probabilities by simply summing up the probabilities over symbols that have common bit values.

These bit probabilities are used as *a priori* probabilities in (5.3). These new *a priori* probabilities replace the third term in (5.3). The bit-level decoder then repeats the decoding process and extrinsic information is passed to the symbol-level decoder. This decoding loop keeps repeating until convergence, hence the word "iterative". Note again that the convergence is not guaranteed. Experimentally, we keep the iteration until there is no further improvement in the cost function.

5.3 Experiment

This section describes how the experiments were set up and how the results were obtained.

We generated data by synthesizing 64,000 first-order Gauss-Markov random samples, with a correlation coefficient of 0.9. A scalar quantizer with four uniform quantization levels is used as a source (symbol-level) encoder. For the channel (bit-level) encoder, a binary convolution code with a rate of $1/2$, and encoder generators $G_1 = 5$ and $G_2 = 7$ is used (Figure 5.5).

As usual, we modelled a noisy channel as a binary symmetric channel with a specific channel bit error rate. Each encoded bit is passed to this discrete channel. The channel bit error rate ranges from 2×10^{-1} to 10^{-2} were used in this simulation.

On the decoder side, the channel decoder uses the BCJR algorithm in (5.11) to obtain the *a posteriori* bit probabilities. After converting to symbol probabilities according to (5.8), the symbol extrinsic information is extracted from these probabilities, before feeding into a soft-decision SAMMSE decoder.

Since a four-level scalar quantizer is used in the encoder, the number of states for the soft-decision SAMMSE decoder is 4. The states in SAMMSE correspond to the quantization levels. However, it does not generate a hard output. The soft-decision SAMMSE decoder produces the *a posteriori* symbol probabilities, at each symbol time, which are then used (at convergence) in a conditional mean estimator to produce the reconstructed symbols by using:

$$\hat{s_t} = \sum_{l=0}^{N-1} y(l) P[I_t = l | \underline{R} = \underline{r}], \quad \forall t, \tag{5.13}$$

where N is the number of quantization levels, 4 in this case, and $y(l)$ the quantization value corresponding to the lth level.

The reconstructed symbols are used to calculate a signal-to-noise ratio (in dB):

$$SNR(dB) = 10 * \log_{10} \frac{\mathrm{E}\{\mathrm{S}^2\}}{\mathrm{E}\{(\mathrm{S} - \hat{\mathrm{S}})^2\}}, \tag{5.14}$$

where S is the original data, and $\hat{S}$ is the reconstructed data. The result obtained at this point is for the serial case (iteration 0). As described earlier, we can feed back the decoded information. The *a posteriori* symbol probabilities obtained from the second stage decoder are used to obtain extrinsic symbol information. This information is converted back to bit probabilities and fed back to the first stage as *a priori* probabilities.

The SNR (in dB) results are obtained for each iteration, given a fixed channel bit error rate. Ten simulations are executed and an average SNR, for each channel bit error rate, is plotted in the result section. Number of symbols and bits errors are also plotted in the results section, for each channel bit error rate.

5.4 Results

This section discusses the results obtained from the experiments described in the previous section.

Figure 5.6 plots the average SNR (in dB) at various channel bit error rates, for different iterations, for an MMSE decoder. It is obvious that the SNR drops as the noise in the channel increases. For the channel bit error rate less than 2×10^{-2}, there is no gain in SNR after the iterations. For the channel bit error rate greater than 2×10^{-2}, the SNR improves significantly after just one iteration. At 2×10^{-1} BER, the SNR improves almost 0.4 dB with one iteration. After the second iteration, the SNR shows further improvement. At 2×10^{-1} BER, the SNR improves 0.1 dB from the first iteration result. Comparing with no iteration, the SNR gains 0.5 dB with two iterations. However, after the third and fourth iterations, there is no obvious gain in the SNR.

Also, as seen from Figure 5.7, the improvement of the SNR at a higher channel bit error rate (most noise in the channel) is larger than the improvement of the SNR at a lower channel bit error rate (less noisy channel). This is because, at a low noise level, the decoder already performs near its optimum; the iteration will not help much in improving the performance. But at a high level of noise, the result from the first pass of decoding (no iteration) is far below optimum. The information on the transmitted information can be extracted from the decoded sequence. This information helps in improving the performance on subsequence iterations.

Figure 5.7 plots the average SNR (in dB) at various channel bit error rates, for different iterations, for a MAP decoder. As explained earlier, the SNR of a MAP decoder

is usually lower than that of a MMSE decoder. This fact is shown when comparing both Figure 5.6 and 5.7. Similarly to an MMSE decoder performance plot, the SNR of an MAP decoder drops as the noise in the channel increases. For the channel bit error rate less than 2×10^{-2}, there is no gain in SNR after the iterations. For the channel bit error rate greater than 2×10^{-2}, the SNR improves significantly after just one iteration. At 2×10^{-1} BER, the SNR improves almost 1 dB with one iteration. After the second iteration, the SNR shows further improvement. At 2×10^{-1} BER, the SNR improves 0.2 dB from the first iteration result. Comparing with no iteration, the SNR gains 1.2 dB with just two iterations. Unlike an MMSE decoder case, after the third iteration, there is still a small gain in the SNR. At 2×10^{-1} BER, the SNR improves 0.1 dB over the second iteration result. However, any further improvement in the SNR is minimal after the third iteration.

Figure 5.8 shows the number of symbol errors at different channel bit error rates, for different iterations, for an MMSE decoder. It is clear from the plot that the number of symbol errors can be reduced after the first iteration, especially at the channel bit error rate higher than 1×10^{-1}. After four iterations, the number of symbol errors is reduced almost in half.

Figure 5.9 shows the number of bit errors at different channel bit error rates, for different iterations, for an MMSE decoder. The number of bit errors is not a linear function of numbers of symbol errors. It is possible that, in one symbol, there are many bit errors. From Figure 5.9, it looks similar to the symbol errors plot in the sense that the number of bit errors is reduced with the number of iterations, especially at a channel

bit error rate higher than 1×10^{-1}. After one iteration, there is a significant reduction in number of bit errors. After the three iterations, the effect of iteration is very minimal.

5.5 Conclusion

We introduced a new iterative turbo-type joint source-channel decoder in this chapter. The proposed decoder capitalizes on the residual redundancy that typically remains after the encoder. The underlying idea for this type of decoder comes from a parallel turbo decoding technique. The proposed decoder iteratively decodes the corrupted bits with newly updated *a posteriori* probabilities. The first stage of the decoder is a soft *bit-level* channel decoder. The outputs of the first stage, which are probabilities, are passed to the second stage which is a soft *symbol-level* source decoder. The output can be used to make a decision (either hard or soft decision) or can be fed back to the first stage to improve the decoding performance. The decoding process can be iteratively repeated. There is no mathematical proof that the performance will improve with the feedback, nor that algorithm converges with respect to a cost function. However, based on experiments, the cost function (SNR) increases with each iteration. There is no further improvement in the cost function after 4-5 iterations.

Although we studied a fixed-length system, this decoding technique can be used for the case of variable length codes system studied in [7], and [54].

Note that the proposed algorithm in this chapter or in [7] never utilized the dependency between sources, as described in previous chapters. Thus, this can be a research direction to study the effect of capturing of such dependency to improve the turbo decoding developed here.

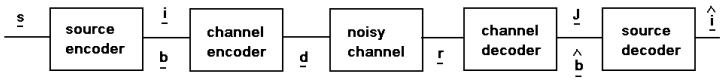

Fig. 5.1. Basic communication system with channel (inner) encoder/decoder. The variables above the line indicate symbol-level variables, while the variables below the line indicate bit-level variables.

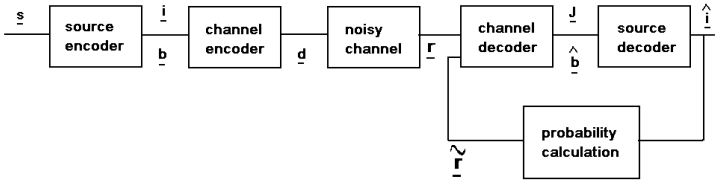

Fig. 5.2. Reference communication system model with feedback decoder that will be studied. The variables above the line indicate symbol-level variables, while the variables below the line indicate bit-level variables.

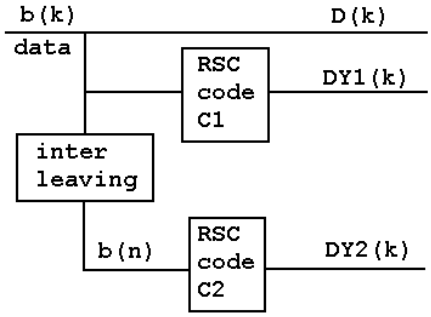

Fig. 5.3. Basic turbo encoder with a rate of 1/3. Both RSC encoders can be different.

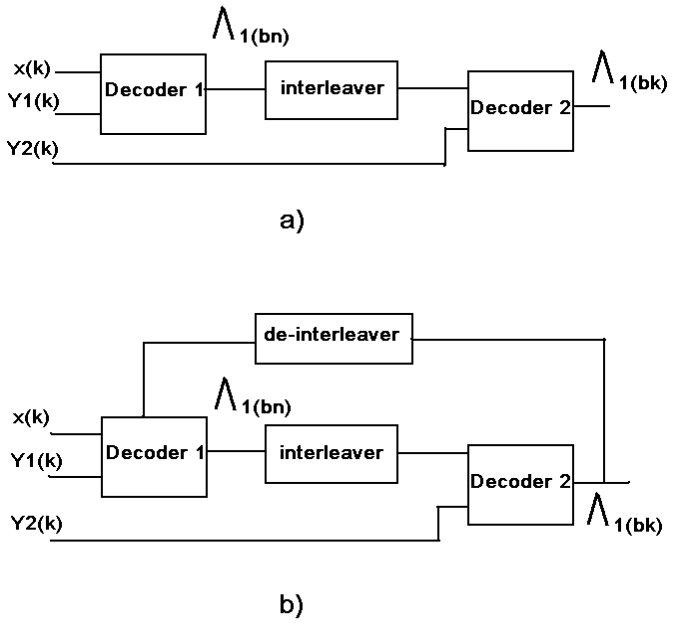

Fig. 5.4. a) Turbo decoder in serial concatenation scheme. b) Turbo decoder in parallel scheme (feedback decoder).

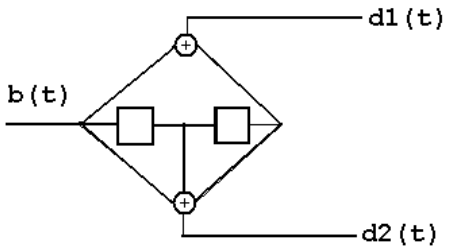

Fig. 5.5. Convolutional encoder with a rate of 1/2 and a [5,7] generator functions.

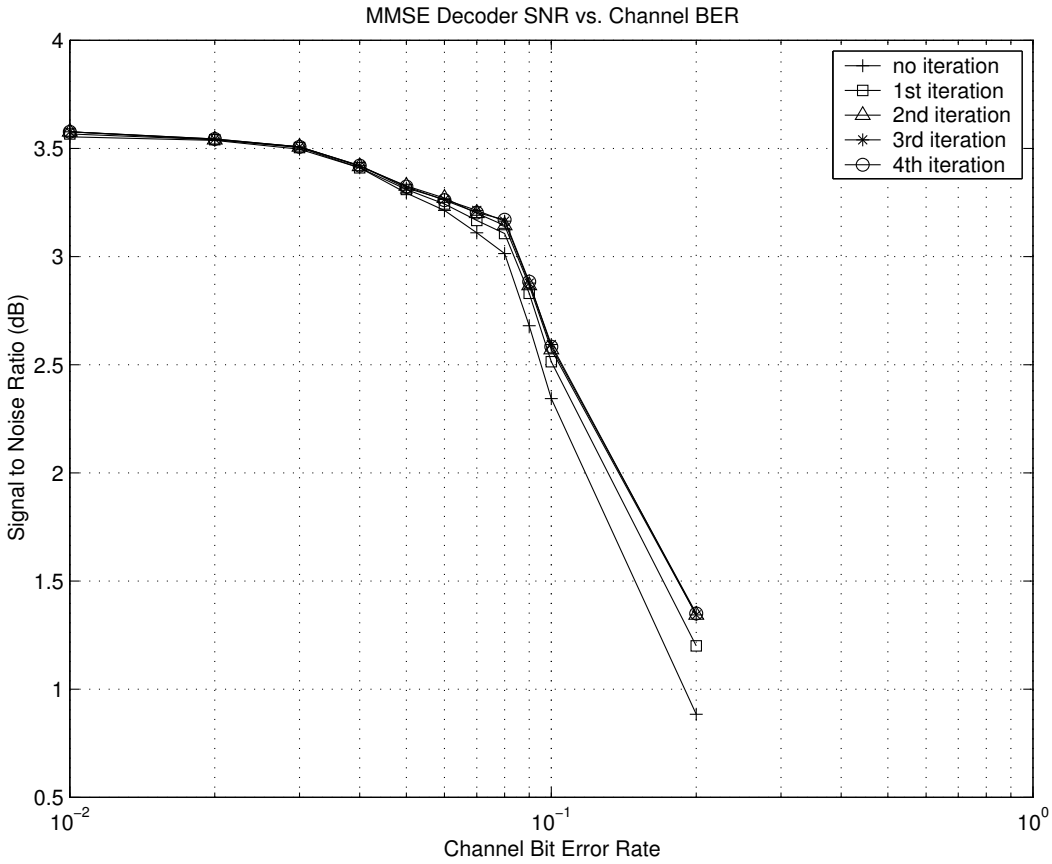

Fig. 5.6. MMSE Decoder SNR (in dB) vs. Channel Bit Error Rate, at different iterations.

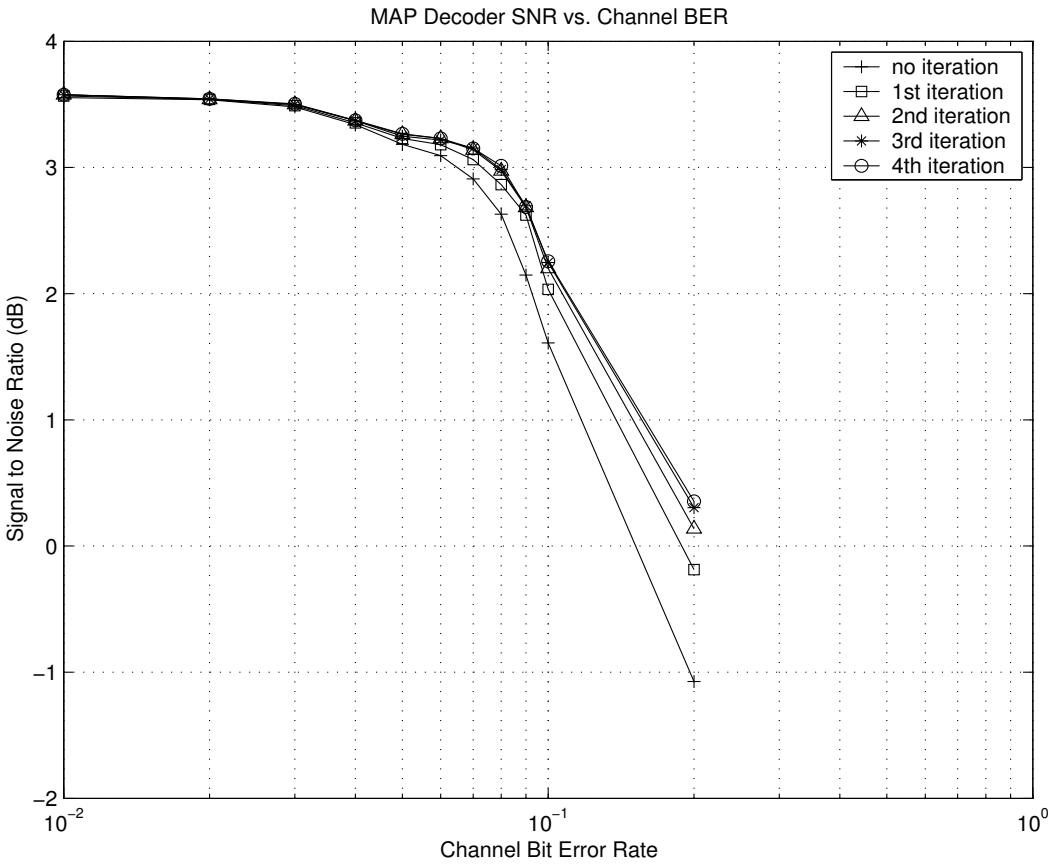

Fig. 5.7. MAP Decoder SNR (in dB) vs. Channel Bit Error Rate, at different iterations.

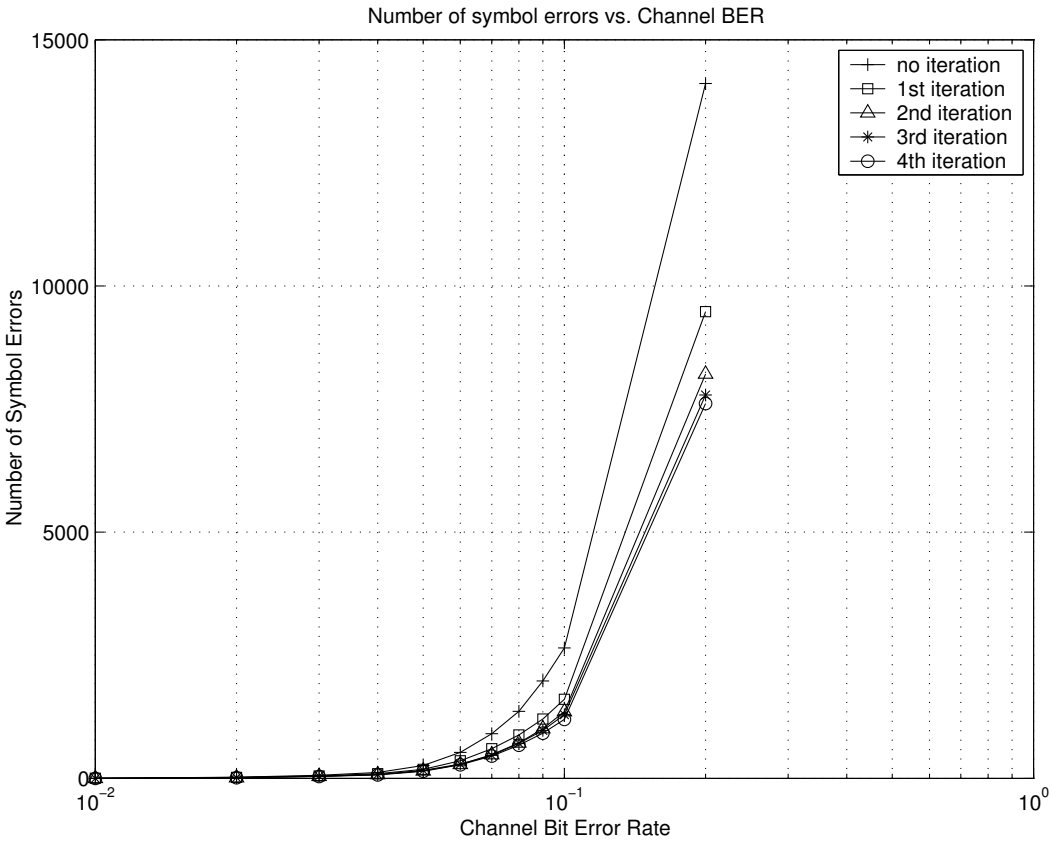

Fig. 5.8. Numbers of Symbol Errors vs. Channel Bit Error Rate, at different iterations.

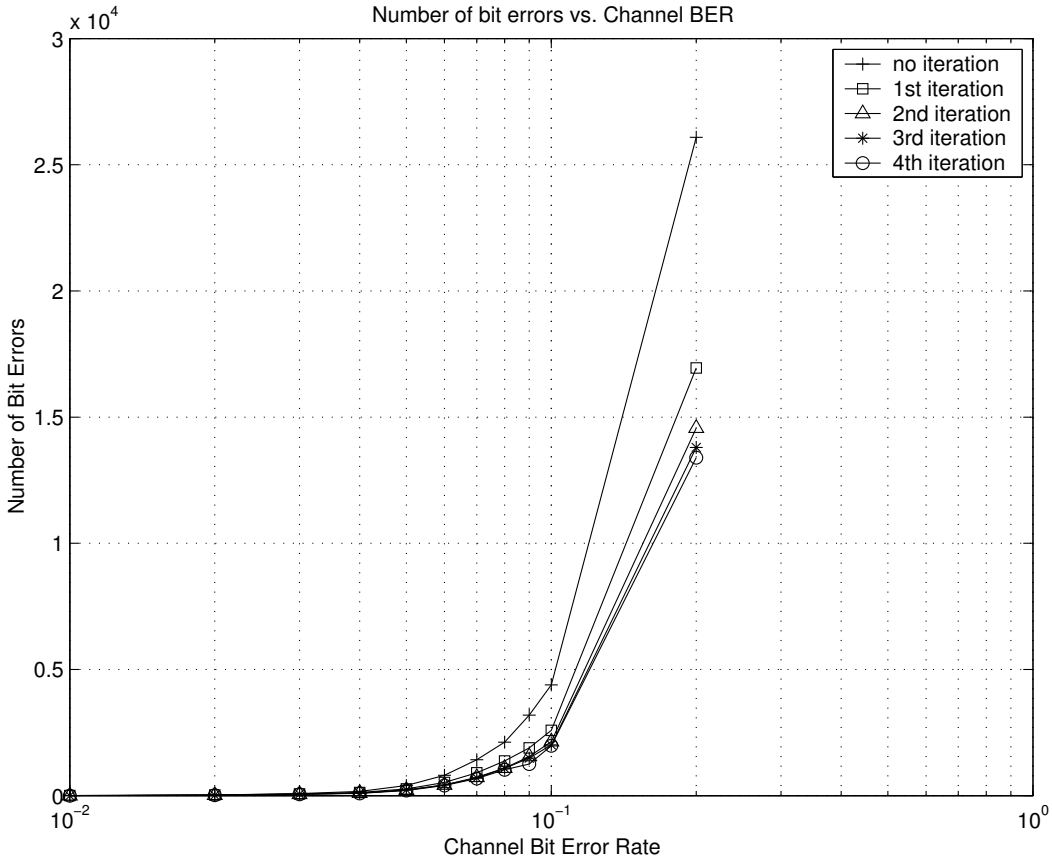

Fig. 5.9. Numbers of Bit Errors vs. Channel Bit Error Rate, at different iterations.

Chapter 6

Iterative Optimization Method for Image Segmentation

6.1 Introduction

In this chapter, we present some ideas on how to apply and extend the iterative optimization technique proposed in this thesis to address the problem of image segmentation. These ideas build somewhat naturally on the methods developed in preceding chapters. The methods described in this chapter represent the results of collaborative works with Prof. David J Miller and Qi Zhao. Our initial work can be found in [52]. A more complete algorithmic description and detailed mathematical developments are presented in [50]. This chapter will indicate clearly how the main results in this thesis (iterative hillclimbing) serve as a starting point for the development of techniques for extending mean-field annealing. Thus, this chapter serves as a bridge between this thesis and some current and continuing work, ongoing in Prof. David J. Miller's laboratory.

The basic idea is similar to previous chapters, where one tries to exploit the redundancy of sources (images) to help assigning a finite set of index values to each pixel site. In image segmentation, the goal is to estimate the value for the unknown (hidden) label at each pixel site in an image, given a (possibly noise-corrupted) observed image.

Markov random fields (MRFs) are random label models that well-characterize spatial smoothness of label segments. MRFs have a 2-dimensional Markov model structure. In image segmentation applications, the HMM states correspond to site labels,

which are unknown. Thus, hidden MRFs are used. Exact state estimation for hidden MRFs is well-known to be intractable. Candidate state estimation approaches range from the computationally intensive (simulated annealing) to the computationally light but suboptimal Iterated Conditional Modes (ICM) [10]. The same iterative hillclimbing idea that was proposed in previous chapters can be modified to tackle this problem as well.

ICM is very closely related to our hillclimbing algorithm in the sense that it iteratively searches for a maximum solution by optimizing over the space of hard assignments. However, ICM searches for a solution pixel by pixel, while our method tries to optimize one row at a time.

Another method that is widely used in this application area is a mean-field annealing (MFA). MFA is a widely used method for optimization grounded in statistical physics. It has been applied to graph-based optimization [11], image segmentation [31], image reconstruction [25], as well as a host of other problems. In recent years, several authors have investigated improved mean field approaches e.g. [37]. There has also been recent work on variational methods which includes MFA as a special case [33]. One can think of MFA as an optimization task involving the determination of a set of discrete-valued assignment variables. MFA searches over a space of probability distributions for assigning labels to sites. After finding optimal probabilities, a MAP rule is used to threshold them to get *hard* assignments. One way of deriving MFA is via maximum entropy, where one seeks the joint distribution over the (random) assignments subject to an average level of cost. MFA is obtained by assuming the individual assignments are independent.

The iterative hillclimbing algorithm that was proposed in previous chapters has been studied in [6] for image segmentation applications. The simulation results show that the cost function values achieved are comparable to the result obtained from an MFA method. However, the segmentation error (misclassification) rate is still higher than that from MFA. This motivated us to attempt to develope a new iterative algorithm that borrows features from both iterative hillclimbing and MFA.

In this chapter, we discuss an MFA extension [50],[51],[52] for problems defined on the pixel sites of an image. Rather than introducing variables for individual sites, we represent label choices *for an entire image row (or column).* We then make the less restrictive assumption of independent *row* (rather than pixel) labeling. While it is not possible to explicitly evaluate the row labeling distribution, we *can*, via a Forward/Backward algorithm, explicitly evaluate *sums* over this distribution, to obtain *a posteriori* probabilities at individual sites. It turns out that the site probabilities, *in turn*, determine (updated) row labeling probabilities. Thus, the Forward/Backward algorithm forms the basis of an iteration, applied to the rows(columns) of the image, that yields optimized *a posteriori* site probabilities. This method is applied to segmentation of synthetic, noise-corrupted Markov random field images. It will be shown that this method achieves reduced misclassification rates, compared with both ICM and standard MFA.

In section 6.2, a review of an iterative hillclimbing algorithm, ICM and MFA will be given. A formulation of our generalized MFA will also be given. In section 6.3, a detail of the simulation and experiment will be described. Experiment results and a conclusion will be given in 6.4 and 6.5, respectively.

6.2 Formulation

This section gives mathematical formulations on the iterative hillclimbing image segmentation technique, ICM, MFA, and the new proposed extension on MFA.

6.2.1 Iterative Hillclimbing Algorithm

[6] investigates the application of our general purpose iterative hillclimbing based algorithm proposed in chapter 2 to the problem of optimizing cost functions for image segmentation applications.

Consider an $N \times N$ gray scale image with sites denoted $S = \{(x,y), x = 1,\ldots N \; y = 1,\ldots,N\}$ and gray scale values $\{g(x,y)\,\forall x,y\}$. Define the *label set* $\mathcal{L} = \{1,\ldots,L\}$, with the random label at (x,y) denoted $I(x,y)$, its realization given by $i(x,y)$, and where $\underline{i} = \{i(x,y)\,\forall x,y\}$. We do not observe this "clean" image, but rather the noisy one $\{z(x,y) = g(x,y) + v(x,y)\,\forall x,y\}$, where $v(x,y)$ is the realization of a zero mean additive white Gaussian noise variate with variance σ^2. The MRF potential to be minimized in choosing a labeling takes the form

$$\begin{aligned} U(\underline{i}) \;=\; & -\sum_{(x,y)\in\mathcal{S}} \Big(\log f(z(x,y)|i(x,y)) + \gamma_{i(x,y)} \\ & + \frac{\mu}{2} \left[\sum_{(l,m)\in\mathcal{N}(x,y)} \delta(i(l,m) - i(x,y)) \right] \Big). \end{aligned} \tag{6.1}$$

Here $f(\cdot)$ is a Gaussian density, $\gamma_{i(x,y)}$ is a prior term, $\mu > 0$ is the MRF "bonding" variable, and $\delta(\cdot)$ is the Kronecker delta. Also, $\mathcal{N}(x,y)$ is a *neighborhood function*. We will assume second order neighbors, i.e. for (x,y) not on the border, all eight surrounding pixels (Figure 6.1).

The algorithm starts from the first row ($y = 1$), searching for the optimal solution (label indices) by using dynamic programming. The obtained solution is optimal in the sense of its row-wise cost. Then, the algorithm moves to the next row. Using dynamic programming and the updated label from previous rows, it searches for the optimal solution for this row's row-wise cost. The algorithm keeps moving until the final row. When the final row is finished, the algorithm goes back to the first row and redoes all the steps. Similar to chapter 2, the algorithm improves the cost function (increases the energy function for image segmentation applications) in a hillclimbing fashion.

This iterative hillclimbing algorithm is closely related to the iterated conditional modes (ICM) algorithm which will be explained next.

6.2.2 Iterated Conditional Modes Formulation

The iterated conditional modes (ICM) algorithm [10] is a popular image segmentation algorithm that has a low computational complexity. ICM finds a maximum *a posteriori* solution by assigning a label to a pixel which has the maximum conditional probability given the observations and the current labels of the site in the neighborhood of the pixel.

At each pixel, the potential to be minimized by the ICM algorithm is

$$\begin{aligned} U(i(x,y)) &= -\log f(z(x,y)|i(x,y)) + \gamma_{i(x,y)} \\ &\quad + \frac{\mu}{2}\left(\sum_{(l,m)\in\mathcal{N}(x,y)} \delta(i(l,m) - i(x,y))\right). \end{aligned} \tag{6.2}$$

Thus, whereas our hillclimbing method optimizes a row-wise cost for each row, ICM optimizes a pixel-wise cost for each pixel. ICM searches for the label $i(x, y)$ that minimizes (6.2). To obtain the label for the whole image, ICM iteratively finds the solution for all x and y. In practice, a raster scan is implemented for updating the labels one by one.

The difference between ICM and the iterative hillclimbing algorithm is that ICM searches for a solution one pixel at a time, while the iterative hillclimbing algorithm searches for a solution one row at a time. In each case one is, at each step, optimizing part of the overall cost (6.1).

Note that both the iterative hillclimbing algorithm and ICM search for site labels at each optimization step. One can think of it as making a hard decision. One might find a better solution if, instead of making hard decisions, a probability distribution for labels at each pixel is calculated. This is what is done in the mean-field annealing algorithm.

6.2.3 Standard MFA Formulation

The standard MFA formulation for image segmentation problems involves the introduction of discrete-valued variables representing the label choices at each pixel site and the *randomization* of these variables, yielding a distribution over the labels. MFA optimization then involves iterative re-estimation of these probabilities, with the re-estimation equations obtained via the *mean-field approximation*, i.e. by assuming labels at the different sites are independent, e.g. [31]. After convergence, a hard decision is finally made to obtain a label for each pixel. One drawback of a MFA method is its computational complexity.

The MRF potential to be minimized in choosing a labeling takes the similar form as (6.1). The MFA approach is simply formulated via the maximum entropy principle [48]. Consider the joint distribution over labels for the entire image, $P[\underline{I} = \underline{i}]$. Note that for brevity's sake, we will omit explicit indication of the dependence on the observations $\{z(x,y)\}$ in all the probabilities we introduce. The *mean-field approximation* amounts to assuming this distribution factors as a product of marginal pmfs over the individual pixel sites. For this case, the joint entropy is a sum of site-wise entropies, i.e

$$H = -\sum_{(x,y)\in\mathcal{S}}\sum_{k\in\mathcal{L}} P[I(x,y)=k]\log P[I(x,y)=k]. \tag{6.3}$$

Moreover, the average MRF potential is

$$\begin{aligned} U &= -\sum_{(x,y)\in\mathcal{S}}\sum_{k\in\mathcal{L}} P[I(x,y)=k]\Big(\log f(z(x,y)|k)+\gamma_k \\ &\qquad +\frac{\mu}{2}\left[\sum_{(l,m)\in\mathcal{N}(x,y)} P[I(l,m)=k]\right]\Big). \end{aligned} \tag{6.4}$$

In MFA, we minimize the free energy $F = U - TH$ over $\{P[I(x,y)=k]\}$ subject to constraints ensuring we obtain valid probabilities. Taking partial derivatives of the associated Lagrangian cost and setting to zero leads to necessary optimal conditions that are also known as *mean field equations*:

$$\begin{aligned} P[I(x,y)=k] &= \frac{\exp\left(\frac{1}{\mathrm{T}}(\log \mathrm{f}(\mathrm{z}(\mathrm{x},\mathrm{y})|\mathrm{k})+\gamma_{\mathrm{k}}+\mu\sum_{(\mathrm{l},\mathrm{m})\in\mathcal{N}(\mathrm{x},\mathrm{y})}\mathrm{P}[\mathrm{I}(\mathrm{l},\mathrm{m})=\mathrm{k}])\right)}{Z(x,y)}, \\ &\quad \forall k,x,y \end{aligned} \tag{6.5}$$

Here, $Z(x,y)$ is the *partition function*, the proper normalization ensuring $P[I(x,y)]$ is a pmf. Note that the equations (6.5) are nonlinear and coupled. These equations form the basis for *fixed point iterations* (FPAs) used to minimize F at fixed "temperature" T [31].

6.2.4 A Sequence-based Extension of MFA

In this work, we demonstrate how an extension and improvement of MFA developed in [50],[51],[52] naturally evolves from the iterative hillclimbing algorithm proposed in this thesis. This approach thus exploits a 1D "structure" existent in certain combinatorial optimization and inference tasks.

We next review the developments in [50],[51],[52]. Rather than explicitly introducing label variables for each *site*, we instead exploit the 1-D (row/column) structure of the image support, introducing variables that represent the *sequence of labelings for an entire row*. Randomizing these, we obtain a distribution over all possible row labelings, for each row in the image. The resulting generalized MFA algorithm has the following properties, to be seen in the sequel:

- The mean-field approximation now amounts to (less restrictively) assuming independent *row*, rather than independent pixel, labelings.

- In standard MFA, the randomized field variables (probabilities at each site) are *directly* re-estimated. However, in our generalized MFA approach, although we can write down re-estimation equations for the row labeling distributions, it is utterly infeasible to actually implement this re-estimation. Even so, we can, in fact,

without approximation, re-estimate quantities that *determine* the row labeling distributions at each iteration. These quantities are the *a posteriori* label probabilities at each pixel site.

- The re-estimation is based on the Forward/Backward (F/B) algorithm, more typically applied for inference in hidden Markov models.

- Ultimately, the converged *a posteriori* probabilities at each site are in fact the quantities of interest, needed to implement a maximum *a posteriori* (MAP) segmentation.

While MFA optimizes over the space of joint image labeling distributions $P[\underline{I} = \underline{i}]$, the optimization is restricted to those distributions consistent with statistically independent labels. If the optimization were instead over a *less* restrictive class of distributions, a better solution might be found. Accordingly, suppose the joint labeling distribution factors as a product over the *rows* of the image, i.e. suppose the random *vectors* of labels for distinct rows are independent.

Consider the random vector of labels for row x, i.e. $\underline{I_x} \equiv (I(x,1), I(x,2), \ldots, I(x,N))$, with realization $\underline{i_x} \in \mathcal{L}^N$. Then, assuming independent row label vectors, the joint entropy is

$$H = -\sum_{x=1}^{N} \sum_{\underline{i} \in \mathcal{L}^N} P[\underline{I_x} = \underline{i}] \log P[\underline{I_x} = \underline{i}] \tag{6.6}$$

and the MRF potential is

$$U = \sum_{x=1}^{N} \sum_{\underline{i} \in \mathcal{L}^N} P[\underline{I_x} = \underline{i}] U_x(\underline{i}) \tag{6.7}$$

where

$$U_x(\underline{i}) = -\sum_{y=1}^{N}\Big(\log f(z(x,y)|i(x,y)) + \gamma_{i(x,y)} + \frac{\mu}{2}\left[\sum_{(l,m)\in\mathcal{N}(x,y)}\sum_{\underline{i_l}\in\mathcal{L}^N : i(l,m)=i(x,y)} P[\underline{I_l} = \underline{i_l}]\right]\Big). \tag{6.8}$$

Necessary conditions for minimizing the free energy $F_{\text{seq}} = U - TH$ subject to constraints ensuring $P[\underline{I_x}]$ is a pmf, $\forall x$, are again given by *mean-field equations*:

$$P[\underline{I_x} = \underline{i}] = \frac{\exp\left(-\frac{1}{\mathrm{T}}\mathrm{U_x(\underline{i})}\right)}{\sum_{\underline{i'}\in\mathcal{L}^N}\exp\left(-\frac{1}{\mathrm{T}}\mathrm{U_x(\underline{i'})}\right)}, \quad \underline{i}\in\mathcal{L}^N, \forall x. \tag{6.9}$$

In principle, (6.9) forms the basis for FPAs by which one could minimize F_{seq} over $\{P[\underline{I_x}],\ \forall x\}$. In particular, the update at the $(t+1)$st iteration would be:

$$P[\underline{I_x} = \underline{i}]^{(t+1)} = \frac{\exp\left(-\frac{1}{\mathrm{T}}\mathrm{U_x^{(t)}(\underline{i})}\right)}{Z^{(t)}(x)}, \forall \underline{i}\in\mathcal{L}^N,\ x = 1,\ldots,N \tag{6.10}$$

with the superscript in $U_x^{(t)}(\cdot)$ indicating the potential is evaluated based on $\{P[\underline{I_{x-1}}]^{(t+1)},\ P[\underline{I_x}]^{(t)},\ P[\underline{I_{x+1}}]^{(t)}\}$. Note that this update needs to be specialized at the border rows of the image. (6.10) indicates a *sequential* procedure, applied row-by-row, with multiple passes (iterations) taken over the image. This is clearly practically infeasible – the FPAs (6.10) will require computation and storage for NL^N values ! Moreover, ultimately, we are not really interested in $P[\underline{I_x}]$ but rather in the *a posteriori* label pmfs at each *site*, $P[I(x,y)]$. These can be used directly for MAP labeling at each site.

While (6.10) cannot be implemented *directly*, we next show that (see [50],[51],[52] for a more detailed development and mathematical validation) it can be implemented *indirectly*, i.e. we develop a re-estimation procedure which, at each iteration, produces quantities that *determine* the row labeling probabilities that *would have been obtained* had we taken an iteration of (6.10). Quite conveniently, it turns out that these quantities are nothing but the desired site pmfs $\{P[I(x,y)]\}$. Thus, we next develop a procedure *equivalent* to the FPAs (6.10), that yields, both at each iteration and at convergence, optimized site pmfs consistent with the optimized row labeling pmfs.

We first simply note that the site pmfs are given by sums over the row pmfs, i.e.

$$P[I(x,y)=k] = \sum_{\underline{i}\in\mathcal{L}^N : i_y = k} P[\underline{I_x} = \underline{i}]. \tag{6.11}$$

Moreover, in (6.10), looking at the form of $U_x(\cdot)$, we see that $P[\underline{I_x}]^{(t+1)}$ depends on the row pmfs $\{P[\underline{I_{x-1}}]^{(t+1)}, P[\underline{I_x}]^{(t)}, P[\underline{I_{x+1}}]^{(t)}\}$ *only* through sums such as $\sum_{\underline{i}\in\mathcal{L}^N : i_y = k} P[\underline{I_x} = \underline{i}]^{(t)}$. But this is just $P[I(x,y)=k]^{(t)}$. I.e., the site pmfs $P[I(x,y)]$ *determine* updated row pmfs. Thus, in order to determine $P[\underline{I_x}]^{(t+1)}$ for some (nonborder) row x, all we need to do is evaluate $\{P[I(x-1,y)]^{(t+1)}, P[I(x,y)]^{(t)}, P[I(x+1,y)]^{(t)}\ \forall y\}$ based on (6.11). These site pmfs can in fact be efficiently, precisely computed via the F/B algorithm of HMM fame. In particular,

$$P[I(x,y)=k]^{(t+1)} \equiv \sum_{\underline{i}\in\mathcal{L}^N : i_y = k} P[\underline{I_x} = \underline{i}]^{(t+1)} = \frac{\alpha_{(x,y)}(k)\beta_{(x,y)}(k)}{\sum_l \alpha_{(x,y)}(l)\beta_{(x,y)}(l)}, \tag{6.12}$$

where α's at iteration $t+1$ are given by the *forward* recursion:

$$\begin{aligned}\alpha_{(x,y)}(k) &= \sum_{n=1}^{L} \alpha_{(x,y-1)}(n)\exp\Big\{\frac{1}{T}\Big(\log f(z(x,y)|k) \\ &\quad +\gamma_k + \mu \sum_{(l,m)\in\mathcal{N}(x,y):l\neq x} P[I(l,m)=k] \\ &\quad +\delta(n-k)\mu P[I(x,y-1)=k]\Big)\Big\} \quad \forall\ k,\ y=1,\ldots,N, \end{aligned} \tag{6.13}$$

initialized by $\{\alpha_{(x,0)}(k)=1\ \forall k,x\}$. The β's are given by the *backward* recursion:

$$\begin{aligned}\beta_{(x,y)}(k) &= \sum_{n=1}^{L} \beta_{(x,y+1)}(n)\exp\Big\{\frac{1}{T}\Big(log f(z(x,y+1)|n) \\ &\quad +\gamma_n + \mu \sum_{(l,m)\in\mathcal{N}(x,y+1):l\neq x} P[I(l,m)=n] \\ &\quad +\delta(n-k)\mu P[I(x,y)=k]\Big)\Big\} \quad \forall k,\ y=N,\ldots,1, \end{aligned} \tag{6.14}$$

initialized by $\{\beta_{(x,N+1)}(k)=1\ \forall k,x\}$.

Thus, at each iteration, we simply re-estimate the site pmfs for each row (visited sequentially) using F/B, which determines the re-estimated row pmfs. The resulting F/B-based MFA optimization algorithm is described in pseudocode as follows:

Forward/Backward-based Mean-field Annealing Algorithm:

1. `Initialize` $\{P[I(x,y]\ \forall x,y\}$ `via uniform pmfs`; $t \leftarrow 0$

2. `Do`

   - `Apply F/B to row 1 given` $\{P[I(1,y)]^{(t)}, P[I(2,y)]^{(t)}\ \forall y\}$, `yielding` $\{P[I(1,y)]^{(t+1)}\ \forall y\}$.
   - `For` $x = 2, \ldots, N-1$
     - `Apply F/B to row` x `given` $\{P[I(x-1,y)]^{(t+1)}, P[I(x,y)]^{(t)}, P[I(x+1,y)]^{(t)}\ \forall y\}$, `yielding` $\{P[I(x,y)]^{(t+1)}\ \forall y\}$.
   - `End For`
   - `Apply F/B to row` N `given` $\{P[I(N-1,y)]^{(t+1)}, P[I(N,y)]^{(t)}\ \forall y\}$, `yielding` $\{P[I(N,y)]^{(t+1)}\ \forall y\}$.
   - $t \leftarrow t+1$

3. `repeat step 2 until converged`

Each image pass *determines* the next iteration of (6.10). Convergence can be based, e.g. on diminishing changes in the site pmfs, from one iteration to the next.

6.3 Experiment

This section explains how the simulations were set up and how the results were obtained.

We performed experiments on twenty-five 80x80 three-class images generated (via Gibbs sampling) [38] based on the MRF model. The three (equally likely) classes had

constant gray levels of 64, 128, and 192, respectively. All images were synthesized with $\mu = 0.7$ and both $\sigma = 64$ and $\sigma = 96$. These values were also used for segmenting images. Moreover, we fixed the temperature, $T = 1$, both for image synthesis and for the MFA algorithms.

After synthesizing all 25 images, different segmentation methods that we applied to the images are listed below:

- ICM.
- serial MFA.
- row-only sweeps serial proposed MFA.

The simulations were run up to 200 iterations (image sweeps). Note that each iteration for each segmentation method requires different computational power. The simulation time will be mentioned in the result section.

Some test images and results from different methods are shown in result section. Misclassification rates (averaged over all 25 images) versus the number of segmentation sweeps through the image are shown in Figure 6.3 and Figure 6.4, for σ equals 64 and 96, respectively.

6.4 Results

A visual inspection on Figure 6.2 shows that the result obtained from ICM method is not quite accurate. The result from a serial MFA looks much better. However, the result is too clean. The MFA smooths the image so most small regions are removed from

the resulting image. The new, sequence-base MFA shows visual improvement in several regions of the image.

Figure 6.3 and Figure 6.4 show the misclassification ratio (averaged over 25 images) for σ equals 64 and 96, respectively. All the methods appear to converge at or before $\sim$ 200 image sweeps.

At convergence, both serial MFA and the newly proposed MFA methods are superior to ICM, but the new MFA methods are substantially better than MFA. ICM requires the least computation while the new MFA method requires the most. For a non-optimized MATLAB implementation of all the algorithms, run on a Sparc5 workstation, ICM, serial MFA, and the new MFA method required $\sim$ 4.7, 7.7, and 14.9 seconds per image sweep, respectively. However, no amount of additional computation would allow ICM or serial MFA to achieve the performance at convergence of the new MFA method. The performance advantage of our methods appears to grow with the noise level.

6.5 Conclusion

Although the iterative hillclimbing optimization technique developed in previous chapters shows a promising result in other applications, [6] shows that the iterative hillclimbing optimization technique does not give a competitive result comparing to other image segmentation techniques, especially a MFA algorithm. This motivated us to development of a sequence-based MFA. With some modification on the standard MFA algorithm [50],[51],[52], the segmentation error improves dramatically. Specifically, the dynamic programming (Viterbi) sweeps are replaced by Forward/Backward sweeps.

This algorithm can be generalized to other applications as well [51]. A more detailed study and development of sequence MFA is a subject of future work.

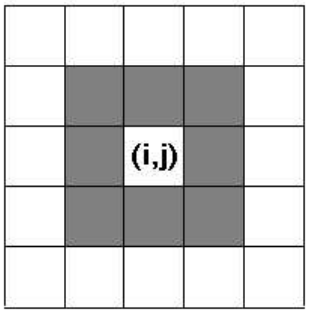

Fig. 6.1. Second order neighborhood of pixel (i,j) is represented by the gray area.

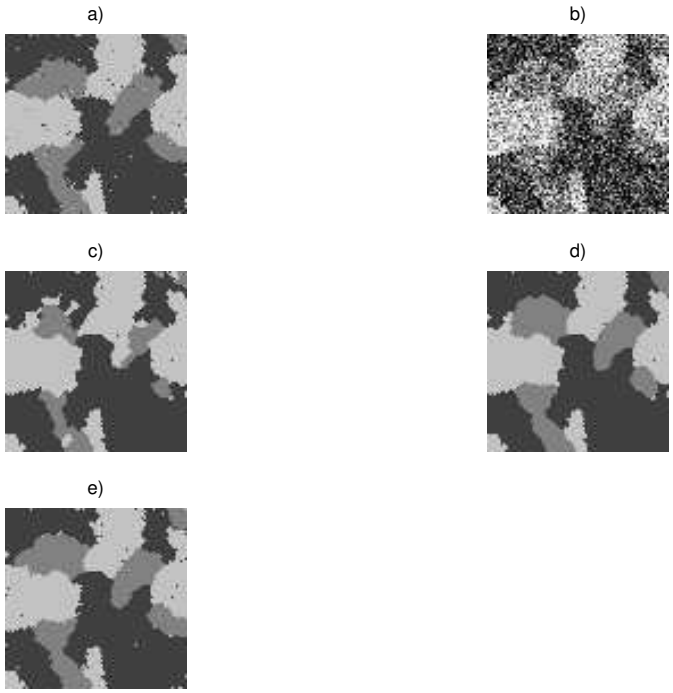

Fig. 6.2. Result: a)Original, b)Noisy (σ=96), c)ICM, d)Serial MFA, e)Row only serial new MFA.

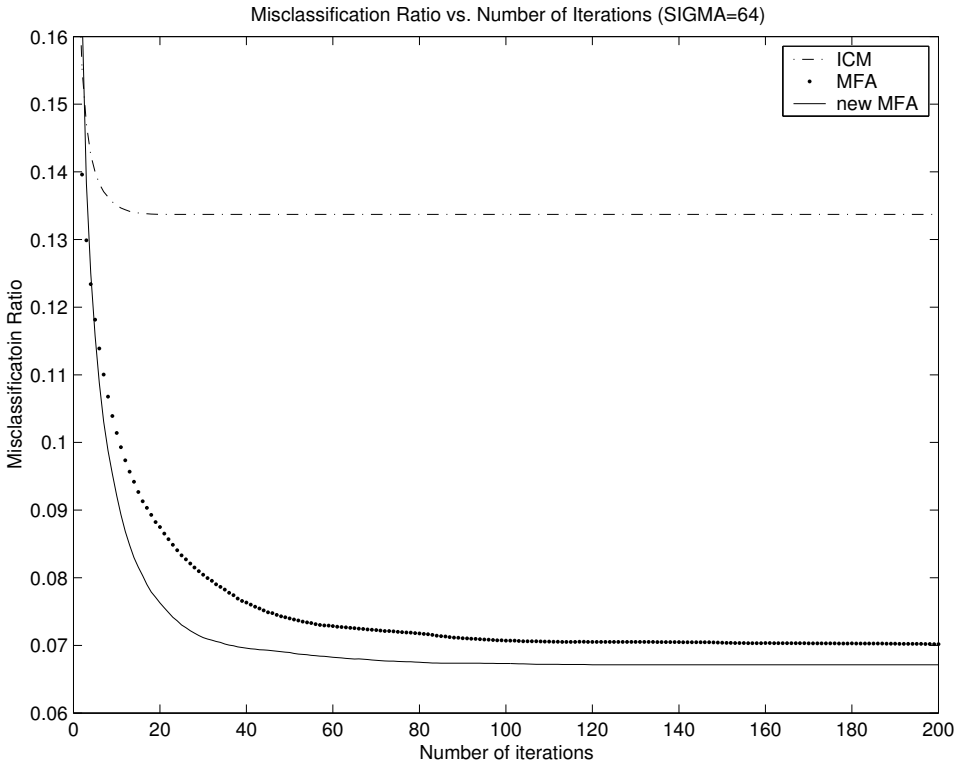

Fig. 6.3. Misclassification ratio for $\sigma = 64$.

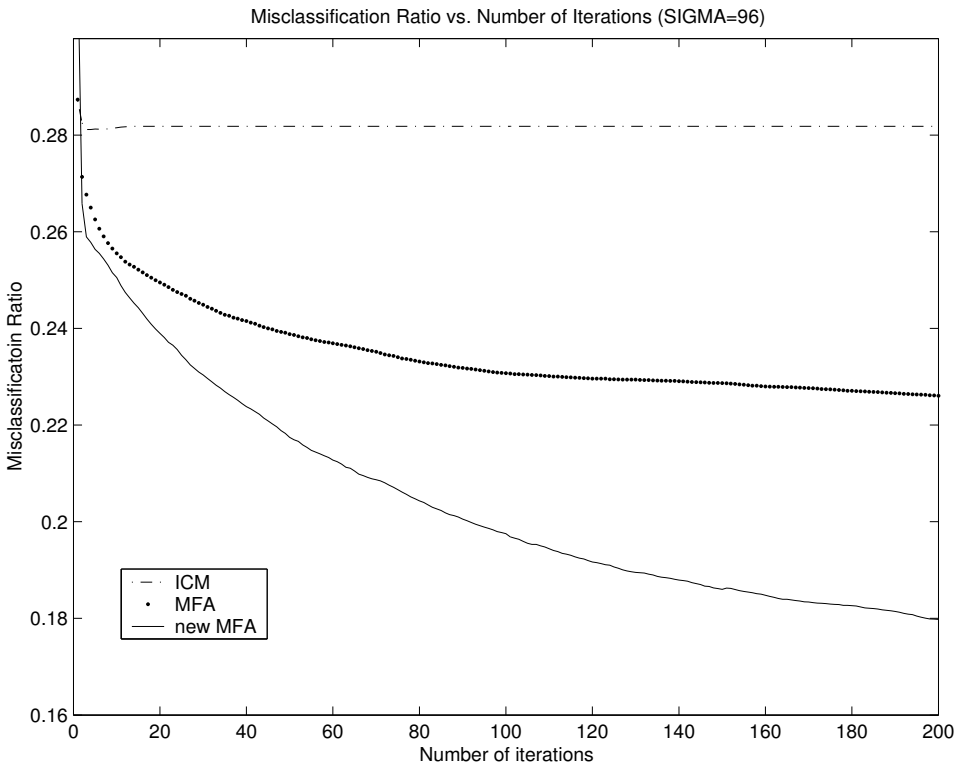

Fig. 6.4. Misclassification ratio for $\sigma = 96$.

Chapter 7

Conclusion

In this thesis, we studied image coding applications in chapter 2. We reviewed several encoding algorithms and used them for result comparison purpose. The new iterative encoding algorithm was proposed to improve the encoding performance. We showed that the proposed algorithm has a hillclimbing property, i.e. it gives a non-increasing cost function after each iteration. Although, the proposed algorithm is not optimal, it is guaranteed to give a performance, at worst, the same as a standard greedy algorithm. From the experiment, it is shown that the proposed algorithm always outperforms the standard greedy algorithm.

In chapter 3, we modified the iterative hillclimbing algorithm and extended it to encode hyperspectral images. Since hyperspectral images can be thought of as a stack of 2-D images, the algorithm proposed in chapter 2 can be straightforwardly applied, with a slight modification to capture 3-D dependencies. However, because of the DPCM used in the system, we can no longer claim the hillclimbing property of the proposed algorithm. Nevertheless, the experimental results show an improvement of 0.9 dB over a standard hyperspectral image encoding algorithm.

After a closer look at the iterative hillclimbing algorithm proposed in chapter 2 and 3, we realized that the algorithm can be generalized to other discrete optimization

problems besides image encoding, for example, noisy image decoding problems and image segmentation problems. This leads to chapter 4 and chapter 6.

We studied noisy image decoding techniques in chapter 4. After reviewing basic decoding techniques, we proposed an iterative hillclimbing algorithm for noisy image decoding to improve decoding performance. Our iterative algorithm has a hillclimbing property on the cost function (likelihood function). The experimental results show an improvement over standard decoding techniques when noise level is high.

In chapter 5, we studied a turbo decoding problem in a joint source-channel decoding system. The algorithm we used is another kind of iterative decoding technique. The decoder iteratively passes soft information between the bit-level and symbol-level decoders to improve the decoding performance. Although the algorithm does not have a hillclimbing property, and is not guaranteed to converge, the experimental results show an improvement after the first few iterations.

In chapter 6, we reviewed work done in [6], where the iterative algorithm was applied to image segmentation problems. [6] shows that the cost function obtained by the iterative algorithm is comparable with the result obtained by a standard mean-field annealing technique. However, the segmentation result from the iterative algorithm is inferior. While trying to improve the performance of standard mean-field annealing, it was discovered that a standard forward/backward algorithm used in an HMM problem can be conveniently used as the basis for an iterative algorithm. This chapter introduces some promising new ideas and acts as a bridge to continuing research work in David J Miller's research group. These ideas have been more seriously pursued and developed elsewhere [50],[51],[52].

Appendix

Proof on Hillclimbing Iterative Algorithm

Theorem: The new algorithm performs *hillclimbing* starting from the greedy solution, with each iteration over the image either decreasing the overall Lagrangian J or, in the worst case, leaving it unchanged.

Proof: Consider the optimization of row m given the remaining rows held fixed. The optimal choice for row m satisfies

$$\begin{aligned}
\underline{i_m^{(t)}} &= \arg\min_{\underline{i_m}} J(\underline{i_1^{(t)}}, \underline{i_2^{(t)}}, \ldots, \underline{i_{m-1}^{(t)}}, \underline{i_m}, \underline{i_{m+1}^{(t-1)}}, \ldots, \underline{i_N^{(t-1)}}) \\
&= \arg\min_{\underline{i_m}} \Big(D_m(\underline{i_m}) + \lambda B_m(\underline{i_m}, \underline{i_{m-1}^{(t)}}) + \lambda B_{m+1}(\underline{i_{m+1}^{(t-1)}}, \underline{i_m}) \Big) \\
&\quad + \Big(D_1(\underline{i_1^{(t)}}) + \lambda B_1(\underline{i_1^{(t)}}) + D_{m+1}(\underline{i_{m+1}^{(t-1)}}) \\
&\quad + \sum_{n=m+2}^{N} [D_n(\underline{i_n^{(t-1)}}) + \lambda B_n(\underline{i_n^{(t-1)}}, \underline{i_{n-1}^{(t-1)}})] \Big) \\
&= \arg\min_{\underline{i_m}} \Big(D_m(\underline{i_m}) + \lambda B_m(\underline{i_m}, \underline{i_{m-1}^{(t)}}) + \lambda B_{m+1}(\underline{i_{m+1}^{(t-1)}}, \underline{i_m}) \Big). \qquad \text{(A.1)}
\end{aligned}$$

Thus, the cost $J(\underline{i_1^{(t)}}, \underline{i_2^{(t)}}, \ldots, \underline{i_{m-1}^{(t)}}, \underline{i_m^{(t)}}, \underline{i_{m+1}^{(t-1)}}, \ldots, \underline{i_N^{(t-1)}})$ is always less than or equal to $J(\underline{i_1^{(t)}}, \underline{i_2^{(t)}}, \ldots, \underline{i_{m-1}^{(t)}}, \underline{i_m^{(t-1)}}, \underline{i_{m+1}^{(t-1)}}, \ldots, \underline{i_N^{(t-1)}})$. In other words, J can be decomposed as a sum of two partial costs, one with *dependence* on the encoding choice for row m and one that is *independent* of this choice. Global minimization of the former partial cost with respect to row m is achieved by dynamic programming and, based on

(A.1), guarantees descent in J. This minimization is also precisely the operation performed on row m in the hillclimbing algorithm, where $1 < m < M$. In a similar fashion, one can show that the minimizations for row 1 and row M are descent steps in J. We can conclude that each iteration over the image reduces J or in the worst case leaves it unchanged. The algorithm thus performs hillclimbing, descending in J starting from the greedy solution. □

References

[1] G. P. Abousleman, M. W. Marcellin, and B. R. Hunt, "Hyperspectral Image Compression Using Entropy-constrained Predictive Trellis Coded Quantization," *IEEE Trans. on Image Proc.*, Vol. 6, pp. 566 - 573, April 1997.

[2] G. P. Abousleman, "Compression of Hyperspectral Imagery Using Hybrid DPCM/DCT and Entropy-Constrained Trellis Coded Quantization," *Data Compression Conference*, DCC'95 , pp. 322 - 331, 1995.

[3] Airborne Visible/Infrared Imaging Spectrometer Homepage, http://makalu.jpl.nasa.gov/aviris.html

[4] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. on Inform. Theory*, Vol. 20, pp. 284 - 287, March 1974.

[5] L. E. Baum, and T. Petrie, "Statistical Interference for Probabilistic Functions of Finite State Markov Chains," *Ann. Math. Statist.*, Vol. 37, pp. 1559 - 1563, 1966.

[6] M Bayram, "An Iterative Dynamic Programming Algorithm for Hill-climbing Optimization in Image Segmentation," *Master Thesis, Electrical Engineering, Pennsylvania State University*, 2001.

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

[8] C. Berrou, and A. Glavieux, "Near Optimum Error Correcting Coding And Decoding: Turbo-Codes," *IEEE Trans. on Communication*, Vol. 44 , pp. 1261 - 1271, October 1996.

[9] J. Besag, "Spatial Interaction and the Statistical Analysis of Lattice Systems," *J. Royal Statistic Society*, vol. 36, pp. 192-236, 1974.

[10] J. Besag, "On the Statistical Analysis of Dirty Pictures," *J. Royal Stat. Soc. B.*, Vol. 48, pp. 259-302, 1986.

[11] D. E. Van den Bout, and T. K. Miller, "Graph Partitioning Using Annealed Neural Networks," *IEEE Trans. on Neural Networks*, Vol. 1, pp. 192-203, 1990.

[12] 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*, Vol. 9, pp 46 - 50, February 2002.

[13] P. Bunyaratavej, and D. J. Miller, "Locally optimal joint encoding of image transform coefficients," *Int'l Conf. on Acoustics, Speech, and Sig Proc,* pp 2573 - 2576, June 2001.

[14] P. A. Chou, and T. Lookabaugh, "Conditional Entropy- Constrained Vector Quantization of Linear Predictive Coefficients," *ICASSP*, Vol. 1, pp. 197-200, April 1990.

[15] 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*, Vol. 2, pp. 722-726, October 1998.

[16] J. L. Devore, "A Note on the Observation of a Markov Source Through a Noisy Channel," *IEEE Trans. on Info. Theory*, Vol. 20, pp. 762-764, 1974.

[17] S. Emami and S. Miller, "DPCM Picture Transmission over Noisy Channels with the Aid of a Markov Model," *IEEE Trans. on Image Proc.*, Vol. 4, pp. 1473 - 1479, 1995.

[18] B. R. Epstein, R. Hingorani, aJ. M. Shapiro, and M. Czigler, "Multispectral KLT-wavelet Data Compression For Landsat Thermatic Mapper Images," *Data Compression Conference*, DCC'92 , pp. 200 - 208, April 1992.

[19] N. Farvardin and J. W. Modestino, "Optimum Quantizer Performance for a Class of Non-Gaussian Memoryless Sources," *IEEE Trans. Inform. Theory*, Vol. 30, pp. 485 - 497, 1984.

[20] N. Farvardin, "A Study of Vector Quantization for Noisy Channels," *IEEE Trans. on Info. Theory*, Vol. 36, pp. 799-809, 1990.

[21] N. Farvardin, and V. Vaishampayan "Optimal Quantizer Design for Noisy Channels: An Approach to Combined Source-Channel Coding," *IEEE Trans. on Inform. Theory*, Vol. 33, pp. 827 - 838, 1987.

[22] T. R. Fischer and M. Wang, "Entropy-Constrained Trellis-Coded Quantization," *IEEE Trans. on Info. Theory*, Vol. 38, No. 2, pp. 415-426, March 1992.

[23] G. D. Forney, "The Viterbi Algorithm," *Proc. IEEE.*, Vol. 61, pp. 268 - 278, March 1973.

[24] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering,* Second Edition, Addison-Wesley, 1994.

[25] D. Geiger, and F. Girosi, "Parallel and Deterministic Algorithms from MRF's: Surface Reconstruction," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 13, pp. 401-412, 1991.

[26] D. Geman, and S. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Trans. on PAMI*, Vol. 6, No. 6, pp. 721-741, November 1984.

[27] A. Gersho, "Optimal Nonlinear Interpolative Vector Quantization," *IEEE Trans. on Communication*, Vol. 38, pp. 1285-1287, 1990.

[28] Z. Ghahramani, and M. I. Jordan, "Factorial Hidden Markov Models," *Machine Learning*, Vol. 29, pp. 245-273, 1997.

[29] H. Gish, and N. J. Pierce, "Asymptotically Efficient Quantizing," *IEEE Trans. on Info. Theory*, Vol. 14, pp. 676 - 683, September 1968.

[30] S. Gupta, and A. Gersho, "Feature Predictive Vector Quantization of Multispectral Images," *IEEE Trans. Geosci. Remote Sensing*, Vol. 30, pp. 491 - 501, 1992.

[31] T. Hofmann, J. Puzicha, and J. M. Buhmann, "Unsupervised Texture Segmentation in a Deterministic Annealing Framework," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 20, pp. 803-818, 1998.

[32] S. Jaggi, "An Investigative Study of Multispectral Lossy Data Compression Using Vector Quantization," *Proc. SPIE 1702, Hybrid Image and Signal Processing III*, pp. 238 - 249, 1992.

[33] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "Introduction to Variational Methods for Graphical Models," *Machine Learning*, Vol. 37, pp. 183-233, 1999.

[34] F. Kossentini, W. C. Chung, and M. J. T. Smith, "Conditional Entropy-Constrained Residual VQ with Application to Image Coding," *IEEE Trans. on Image Processing*, vol. 5, No. 2, pp. 311-320, February 1996.

[35] A. Kurtenbach, and P. Wintz, "Quantization for Noisy Channels," *IEEE Trans. on Communication*, Vol. 17, pp. 291-302, 1969.

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

[37] M. A. R. Leisink, and H. J. Kappen. "Learning in Higher Order Boltzmann Machines Using Linear Response," *Neural Networks*, Vol. 13, pp. 329-335, 2000.

[38] S. Z. Li, "Markov Random Field Modeling in Computer Vision," *Springer-Verlag*, Tokyo, 1995.

[39] M. Lightstone, D. Miller, and S. K. Mitra, "Entropy-Constrained Product Code Vector Quantization with Application to Image Coding," *IEEE International Conference on Image Processing*, pp. 623-627, November 1994.

[40] R. Link, and S. Kallel, "Markov Model Aided Decoding for Image Transmission Using Soft-Decision-Feedback," *IEEE Trans. on Image Proc.*, Vol. 9, No. 2, pp. 190-196, February 2000.

[41] F.H. Liu, P. Ho, and V. Cuperman, "Joint Source And Channel Coding Using a Non-linear Receiver," *IEEE Int'l Conferences on Communications*, Vol. 3, pp. 1502-1507, 1993.

[42] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. on Info. Theory*, Vol. 28, pp. 127-135, March 1982.

[43] M. Manohar, and J. C. Tilton, "Progressive Vector quantization of Multispectral Image Data Using a Massively Parallel SIMD Machine," *Data Compression Conference*, DCC'92 , pp. 181 - 190, April 1992.

[44] M. W. Marcellin, and T. R. Fischer, "Trellis Coded Quantization of Memoryless and Gauss-Markov Sources," *IEEE Trans. on Communications*, Vol. 38, No. 1, pp. 82-93, January 1990.

[45] M. W. Marcellin, P. Sriram, and K. L. Tong, "Transform Coding of Monochrome and Color Images Using Trellis Coded Quantization," *IEEE Trans. on Cir. and Sys. for Video Tech.*, Vol. 3, pp. 270 - 276, August 1993.

[46] M. W. Marcellin, "On Entropy-Constrained Trellis Coded Quantization," *IEEE Trans. on Communications*, Vol. 42, No. 1, pp. 14-16, January 1994.

[47] B. Martins, and S. Forchhammer, "Lossless, near-lossless, and refinement coding of bilevel images," *IEEE Trans. on Image Procs.*, Vol. 8, No. 5, pp. 601-613, May 1999.

[48] R. Meir, "On Deriving Deterministic Learning Rules From Stochastic System," *Intl Journal of neural Systems*, Vol 2, pp. 283-289, 1992.

[49] 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 Trans. on Communications*, Vol. 46, No. 2, pp. 222-231, February 1998

[50] D. J. Miller, and Q. Zhao, "A Sequence-based Extension of Mean-field Annealing Using the Forward/Backward Algorithm: Application to Image Segmentation" *submitted to IEEE Trans. on Signal Processing*, 2002.

[51] D. J. Miller, and Y. Wang, "Joint Source-channel Image Decoding Via a Sequence-based Extension of Mean-field Techniques," *submitted to ICIP 2002*, 2002.

[52] D. J. Miller, P Bunyaratavej, and Q. Zhao, "A Sequence-based Extension of Mean-field Annealing Using the Forward/Backward Algorithm" *To appear in ICASSP 2002*, 2002.

[53] 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*, Vol. 4, pp. 289 - 292, October 1997.

[54] M. S. Park and D. J. Miller, "Improved Joint Source-Channel Decoding for Variable-Length Encoded Data Using Soft Decisions And MMSE Estimation," *Data Compression Conference*, DCC'99 , pp. 5441, 1999.

[55] M. S. Park and D. J. Miller, "Joint Source-Channel Decoding For Variable-Length Encoded Data by Exact And Approximate MAP Sequence Estimation ," *IEEE Trans. on Communication*, Vol. 48 , pp. 1 - 6, January 2000.

[56] N. Phamdo and N. Farvardin, "Optimal Detection of Discrete Markov sources over Discrete Memoryless Channels – Application to Combined source-Channel Coding," *IEEE Trans. on Inform. Theory*, Vol. 40, pp. 186 - 193, 1994.

[57] J. G. Proakis, *Digital Communications,* Third Edition, McGraw-Hill, New York, 1995.

[58] L. Rabiner and B. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, New Jersey, 1993.

[59] J. A. Saghri, and A. G. Tescher, "Near-Lossless Bandwidth Compression for Radiometric Data," *Opt. Eng.*, Vol. 30, pp. 934-939, July 1991.

[60] K. Sayood, and J. C. Borkenhagen, "Use of Residual Redundancy In The Design of Joint Source/Channel Coders ," *IEEE Trans. on Communication*, Vol. 39, No. 6, pp. 838 - 846, June 1991.

[61] K. Sayood, H. H. Otu, and N. Demir, "Joint Source/Channel Coding for Variable Length Codes," *IEEE Trans. on Communication*, Vol. 48, No. 5, pp. 787-794, May 2000.

[62] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, Vol. 27, pp. 379-423, 623-656, 1948.

[63] L. C. Stewart, R. M. Gray, and Y. Linde, " The Design of Trellis Waveform Coders," *IEEE Trans. on Communication*, Vol. 30, pp. 702-710, April 1982.

[64] G. Storvik, and G. Dahl, "Lagrangian-based Methods for Finding MAP Solutions for MRF Models," *IEEE Trans. on Image Processing*, Vol. 9, pp. 469-479, 2000.

[65] G. Strang, and T. Nguyen, *Wavelets and Filter Banks,* Wellesley-Cambridge, Massashusetts, 1996.

[66] J. C. Tilton. "Hierarchical Data Compression: Integrated Browse, Moderate Loss, and Lossless Levels of Data Compression," *Proc. Int. Geoscience and Remote Sensing Symp.,* pp. 1651-1654, 1990.

[67] J. C. Tilton, D. Han, and M. Manohar, "Compression Experiments with AVHRR data," *Data Compression Conference*, DCC'91 , pp. 411 - 420, April 1991.

[68] S. G. Wilson, *Digital Modulation and Coding,* Prentica-Hall, New Jersey, 1996.

Vita

Piya Bunyaratavej was born in Bangkok, Thailand on November 21, 1973. He received the B.S. degree in Electronics Engineering, *magna cum laude*, from Assumption University, Thailand, in 1994. He then joined Seagate Technology (Thailand) as a software engineer. In 1996, he received the M.S. degree in Electrical Engineering from the George Washington University, Washington DC. In fall of 1996, he enrolled in the Ph.D. program in Electrical Engineering at the Pennsylvania State University. He joins the MathWorks as a senior DSP applications engineer in July, 2001.

His research interests are in the area of digital signal processing in communication applications, data compression and digital communications. He has been a member of the Institute of Electrical and Electronics Engineers since 1995.