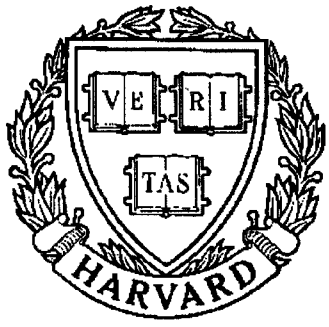
THESIS REPORT
Ph.D.

S Y S T E M S
R E S E A R C H
C E N T E R



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), the University of Maryland, Harvard University, and Industry

Design and Performance Evaluation of a Class of Finite-State Vector Quantizers

by Y. Hussain
Advisor: N. Farvardin

Ph.D. 92-8

ABSTRACT

Title of Dissertation: DESIGN AND PERFORMANCE EVALUATION OF A CLASS OF FINITE-STATE VECTOR QUANTIZERS

Yunus Hussain, Doctor of Philosophy, 1992

Dissertation directed by: Dr. Nariman Farvardin
Associate Professor
Electrical Engineering Department

The finite-state vector quantizer (FSVQ), introduced by Foster, Dunham and Gray, is a finite-state machine that can be viewed as a collection of memoryless full-searched vector quantizers, where each input vector is encoded using a vector quantizer associated with the current encoder state; the current state and selected codeword determine the next encoder state. It is generally assumed that the state codebooks are unstructured and have the same cardinality leading to a fixed-rate scheme. In this thesis, we present two *variable-rate* variations of the FSVQ scheme with the possibility of using structured as well as unstructured state codebooks. In the first scheme, we let the state codebook sizes be different for different states, implying different rate distribution among the states. In the second scheme, in addition to this flexibility, we use pruned tree-structured vector quantizers as the state quantizers, i.e., we let each of the state quantizers be a variable-rate encoder. For

encoding sampled speech data, both of these schemes perform significantly better than the fixed-rate FSVQ scheme with the second scheme giving the best performance.

We also consider the 2-D extension of the above mentioned schemes and describe two low bit rate image coding systems based on these schemes. A compression ratio in excess of 26 is achieved for encoding the 512×512 version of "Lena" using the schemes employing variable-rate FSVQs.

An implicit assumption made in all the systems mentioned above is that the channel is noiseless. Under noisy channel conditions, all of the above systems suffer from severe performance degradations calling for the need to reformulate the FSVQ design problem taking into account the channel noise. Using some of the developments in joint source-channel trellis coding, we describe two different methods leading to two types of noisy channel FSVQs. We show by means of simulations on the Gauss-Markov source and speech LSP parameters and for a binary symmetric channel that both schemes are fairly robust to channel noise. In particular, for speech LSP parameters, the proposed noisy channel FSVQs lead to a saving of 1.5-4 bits/frame over the channel-optimized vector quantizer depending on the level of noise in the channel.

DESIGN AND PERFORMANCE EVALUATION OF A CLASS OF FINITE-STATE VECTOR QUANTIZERS

by

Yunus Hussain

Dissertation submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1992

Advisory Committee:

Associate Professor N. Farvardin, Chairman/Advisor
Professor L. Davisson
Professor E. Geraniotis
Assistant Professor T. Fuja
Professor L. Davis

ACKNOWLEDGEMENTS

I wish to thank my wife, Meena, for her constant encouragement and patience which has helped, in no small way, the completion of this work. This thesis was developed under the guidance of Dr. Nariman Farvardin and I wish to express my gratitude for his invaluable guidance and support and providing the right environment throughout the course of this research. I am also very thankful to all my friends and colleagues in the Communications and Signal Processing Labratory for their help throughout my stay at the University of Maryland, College Park.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data compression can be described as the reduction in the rate to represent an information in comparison with an initial representation. Various data compression techniques can be classified into two categories. The first type consists of all techniques that are noiseless or information lossless, where it is possible to reconstruct the original information perfectly from the compressed data. Examples of such techniques are Huffman coding [1], arithmetic coding [2] and Ziv-Lempel coding [3] among the more prominent ones. The techniques belonging to the second category are referred to as noisy, information lossy or data compression systems with a fidelity criterion and as opposed to the information lossless systems, the original information cannot be perfectly reconstructed from the compressed information in these systems. All of the quantization techniques fall in the second category. The achievable data compression ratio is limited for the noiseless compression systems to single digits, while for the lossy compression systems they can be as high as 80 for still images and several hundred for video compression. In the sequel, we will limit ourselves to compression systems with a fidelity criterion and will focus our attention on a particular type of lossy compression system called the vector quantization (VQ) which is basically an extension of scalar quantization (SQ) to the vector case.

In scalar quantization, a selected set of values of a signal, called the quantization

levels, are allowed to be stored or transmitted. SQ is very commonly used in analog to digital conversion where the signal is possibly continuous-time (in which case, it is first sampled at a fixed sampling period). The sampled signal has continuously varying amplitude which is mapped to the closest quantization level. In scalar quantization, only one value of the signal is quantized at a time while in the VQ, a vector of samples is quantized together to one of the predetermined reproduction vectors.

1.1 Vector Quantization

In the sequel we assume that the source to be encoded is a real-valued, stationary and ergodic process $\mathcal{X} = \{x_n\}_{n=0}^{\infty}$. An L-dimensional N-level VQ for a source $\mathcal{X}$ is in general described by an encoder mapping α and a decoder mapping β. The encoder mapping $\alpha : \mathbb{R}^L \rightarrow \mathcal{N} \triangleq \{1, 2, \ldots, N\}$ ($\mathcal{N}$ is called the channel space) is given by

$$\alpha(\mathbf{x}) = i, \text{ if } \mathbf{x} \in \mathcal{P}_i, \; i \in \mathcal{N}, \tag{1.1}$$

where $\mathbf{x} = (x_{nL}, x_{nL+1}, \ldots, x_{nL+L-1})$ is a block of L successive samples from $\mathcal{X}$ and $\mathcal{P} \triangleq \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$ is a partition of $\mathbb{R}^L$. The decoder mapping $\beta : \mathcal{N} \rightarrow \mathbb{R}^L$ is described in terms of a reproduction codebook $\mathcal{C} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$ according to

$$\beta(i) = \mathbf{y}_i, i \in \mathcal{N}. \tag{1.2}$$

For a given input vector $\mathbf{x}$, its reproduction vector $q(\mathbf{x})$ is given by

$$q(\mathbf{x}) = \beta(\alpha(\mathbf{x})). \tag{1.3}$$

The rate of the VQ is given by $R = \log_2 N$, bits/vector and the performance of VQ is measured in terms of a distortion measure $d : \mathbb{R}^L \times \mathbb{R}^L \rightarrow [0, \infty)$. A VQ is optimal w.r.t d if the average distortion $D = \frac{1}{L} E[d(\mathbf{X}, q(\mathbf{X}))]$ is minimized over all encoder-decoder pairs (i.e. over all $\mathcal{P}$ and $\mathcal{C}$). Two necessary conditions of

optimality [4] are

$$\mathcal{P}_i = \{\mathbf{x} : d(\mathbf{x}, \mathbf{y}_i) \leq d(\mathbf{x}, \mathbf{y}_j), \forall j \in \mathcal{N}\}, i \in \mathcal{N}, \tag{1.4}$$

and

$$\mathbf{y}_i = \arg \min_{\mathbf{y} \in \mathbb{R}^L} E[d(\mathbf{X}, \mathbf{y}) | \mathbf{X} \in \mathcal{P}_i], \forall i \in \mathcal{N}. \tag{1.5}$$

The term $\arg\min_{\mathbf{y}} E[d(\mathbf{X}, \mathbf{y})|\mathbf{X} \in \mathcal{P}_i]$ is called the centroid of $\mathcal{P}_i$. For the squared error, the centroid is just the average of all the vectors in $\mathcal{P}_i$. Based on (1.4) and (1.5), an iterative design algorithm for VQ called LBG algorithm was developed in [4]. The LBG algorithm yields a locally optimum VQ design for a given source or a training sequence and is based on alternately optimizing the encoder (or partition $\mathcal{P}$) and the decoder (or codebook $\mathcal{C}$); an initial decoder is designed first and then the alternating optimization is done for several iterations till convergence occurs. An initial decoder (consisting of N codevectors or reproduction vectors) can be obtained by choosing N vectors from the source or the training sequence randomly. Another way is to design an initial codebook by using the so-called splitting method described in [4]; in fact [4] describes various methods to design an initial $\mathcal{C}$. Once the initial decoder is designed, the encoder is optimized by using the minimum-distortion rule given by (1.4) to partition the training sequence. Next for the new encoder, the decoder is optimized using (1.5). Since we are alternately optimizing the encoder and decoder, the distortion at each step of iteration can only decrease and since the distortion is lower bounded by zero (for squared-error distortion measure), the algorithm has to converge, at least, to a local optimum. The LBG algorithm takes a fair number of iterations to converge and since it involves exhaustive search in (1.4), it is computationally intensive. In fact, the complexity of computation is exponential in R (bits/vector). The VQ designed using the LBG algorithm will be referred to as LBG-VQ in the sequel.

There are other design algorithms for VQ such as pairwise nearest neighbor

algorithm proposed by Equitz [5], which is much faster than LBG algorithm (5% of the CPU time of the LBG algorithm) at an expense of slight increase in distortion. In this algorithm, an initial codebook is chosen to be the training sequence and then two codewords are merged at a time till the desired rate is achieved; the two codewords selected for merging are those which give minimum increase in distortion on merging. Vaishey and Gersho proposed a VQ design algorithm based on simulated annealing [6], which under certain conditions achieves the global optimum (as opposed to local optimum obtained with the LBG algorithm) at the expense of much higher computation.

Since the advent of the VQ and its design algorithms in 1980, the VQ has been used extensively for data compression. The main motivation behind the use of VQ was the result by Shannon [7] that VQ can attain performance close to the "best possible" in the rate-distortion theoretic sense in the limit when the block length goes to infinity. In a practical situation, however, we can only consider finite block lengths, and practical VQ systems fall short of achieving the best, while still performing much better than SQs. The main reason for the superior performance of VQ over SQ is that VQ exploits the correlation between the components of the vector, while SQ does not. Although the performance of VQ at a given rate can be improved by increasing the vector dimension, the resulting complexity places a practical limit on how large a block length can be used. The complexity can be reduced (and hence a larger block length can be used leading to a better performance than the LBG-VQ) by using suboptimal search strategies as in tree-structured VQ (TSVQ) and multi-stage VQ (MSVQ) and special structures as in lattice VQs.

In a TSVQ, a sequence of binary (or higher order depending on the type of tree) comparisons is performed instead of a large exhaustive search. As a consequence, the search complexity increases as $\log N$ instead of N. As depicted in the Figure 1.1, starting at the root node of the tree, if the input vector is closer in min-

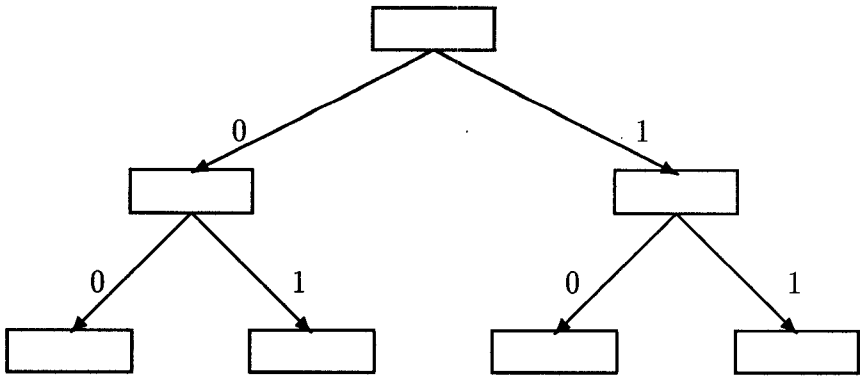


Figure 1.1: A tree-structured vector quantizer.

imum distortion sense to the left child then a '0' is transmitted and we descend to the left child. On the other hand, if the input vector is closer to the right child then a '1' is transmitted and we descend to the right child. We repeat with the selected child till the leaf is reached. Then the next input vector is obtained and the whole process is repeated till the end of the training sequence. The design algorithm for the TSVQ is provided in [8]. Since the TSVQ involves suboptimal search, it suffers, in general, some degradation as compared to the LBG-VQ; in addition, the memory requirements are almost doubled. However, the reduced complexity can more than compensate for both these factors in some of the applications.

MSVQ is a VQ [9] with "successive approximation" characteristics. Instead of using a full-rate (N-level) VQ, the source vector is first approximately quantized using an N_1-level ($N_1 < N$) VQ and then the error of this stage is vector quantized using another N_2-level VQ. The number of stages can be arbitrary in MSVQ. The resulting MSVQ has the advantage of both lower complexity and lower memory. The penalty paid is a higher degradation in performance as compared to the LBG-VQ and even TSVQ.

Lattice VQs [10], [11] are basically L-dimensional generalizations of uniform scalar quantizers and are capable of efficient searches and memory usage. Because

of the special structure of these VQs, there exist algorithmic methods to perform the encoding and decoding operations (without any need to store) for vectors of very large dimensions and VQs of high rates. One problem associated with lattice VQs is that they cannot be iteratively improved using a generalized LBG algorithm without losing their lattice structure in the design process.

All the variations of the LBG-VQ described above are computationally less expensive and hence can be designed for larger dimensions than the LBG-VQ for a given rate in bits/sample leading to a performance improvement over the LBG-VQ. An alternative way to improve the performance of the VQ is to incorporate memory in the VQ structure. Examples of VQs with memory are feedback VQs such as vector predictive VQ (PVQ) and finite-state VQ (FSVQ). Trellis encoding system [12]-[14] can also be classified as VQ with memory and they share the same decoder as the feedback VQ but their encoder is "lookahead" in nature (i.e. they look at several future vectors before encoding the present vector).

The predictive vector quantizer, proposed by Cuperman and Gersho [15] is basically a vector generalization of DPCM. For fixed predictors, a generalized LBG algorithm can be used to design VQ for the residual sequence. Extension of this system is provided in [16] in which a stochastic gradient algorithm is used to iteratively improve the vector linear predictor coefficients.

An FSVQ is a finite-state machine and associated with each state of the finite-state machine there is a VQ. The current input vector is quantized using the VQ associated with the current state; the next state is decided based on the current state and encoder output. Since the encoder determines its next state based on its previous outputs which are also available at the receiver, the decoder can track the encoder state sequence without any extra overhead information. While encoding sampled speech waveform using the FSVQ and PVQ [17], it was observed that the FSVQ performed better. Both these systems have a built-in memory and

therefore can exploit intervector correlation. In the rest of this thesis, we will only concentrate on the FSVQs.

As mentioned earlier, FSVQ is a type of feedback VQ and can be thought of as a time varying VQ. There is a super-codebook that contains a very large number of codevectors and there is an internal state which accurately represents a small region (with a small sized subset of the super-codebook covering this region) that contains the source vector at any given time. Thus FSVQ achieves the efficiency of a large rate codebook at a relatively small rate.

In [18], the FSVQ was introduced and used to encode two sources with memory, namely the 1^{st}-order Gauss-Markov source and sampled speech waveform; performance improvements were observed as compared to the memoryless LBG-VQ. Then in [19], a technique based on adaptive stochastic automata theory was introduced that led to an improved design algorithm for FSVQ and an application of FSVQ was made to voice coding. In [13], Bei and Gray proposed a trellis encoding system which used the decoder of FSVQ [18] and an encoder that was "lookahead" in nature (i.e., their system was same as FSVQ with a "lookahead" encoder); the proposed system was used to encode the Gauss-Markov source, sampled speech waveform and LPC parameters of speech and a gain of up to 0.7-1.0 dB[1] was observed over FSVQ. Several schemes based on FSVQ have also been reported in the image coding literature. In [20] and [21], FSVQ was used to encode still images where the state was used to exploit the correlation in the spatial domain; over 50% saving in bit rate is achieved over LBG-VQ. In [22] and [23], FSVQ was used in coding image sequences where the state is defined to exploit temporal correlation; again a saving of over 50% is achieved over intraframe LBG-VQ.

[1]Distortion measure used for encoding the Gauss-Markov source and sampled speech was squared-error, while the Itakura-Saito distortion measure was used for encoding the LPC coefficients.

1.2 Contribution of the Thesis

In all the above cases involving FSVQs, an implicit assumption made is that the channel over which data is to be transmitted is noiseless. Actually, it is only under noiseless channel conditions that the decoder can perfectly track the encoder state sequence. We will also maintain the assumption of a noiseless channel in the first part of the dissertation. In the FSVQ systems mentioned above, the state codebooks are all assumed to have the same cardinality and hence the same bit rate. This, in a loose sense, implies that all the states are treated with approximately equal degree of fidelity. Such a restriction limits the performance of the FSVQ which has the potential of doing better. In this work, we have relaxed this assumption and have considered an FSVQ scheme with different codebook sizes for different states while constraining the average bit rate. This modification has resulted in performance improvements (in some cases substantial) in terms of signal-to-noise ratio. We have also considered the possibility of using structured VQs such as the tree-searched VQ (TSVQ) [8] and optimally pruned unbalanced TSVQ (UTSVQ) [24] as the state quantizers. In fact, the scheme using UTSVQ as state quantizers and with the flexibility of having variable bit rate assignment among the states performs the best amongst all the schemes considered in this thesis under noiseless channel assumptions, as will be shown by the simulation results. We will use the modified FSVQ systems to encode a composite Gauss-Markov source and sampled speech waveform. Also the extension of various FSVQ systems to 2 dimension (2-D) is considered and various schemes based on 2-D FSVQs are described for encoding still images.

In the second part of the dissertation, the problem of designing FSVQ for the noisy channel case is considered; the channel over which the FSVQ encoder output is transmitted is no longer assumed to be noiseless. We limit ourselves to

fixed-rate FSVQ systems and a discrete memoryless channel (DMC). Under these conditions, the decoder may not be able to track the encoder state sequence leading to a substantial degradation in the performance of FSVQ of [18]. Thus the noisy channel case calls for redesigning the FSVQ structure.

For a continuous alphabet source, the minimum distortion that has to be tolerated if it is transmitted (after source encoding) over a noisy channel is given by the distortion rate function evaluated at the channel capacity [7], where the distortion rate function determines the minimum distortion that must be tolerated in order to represent the source by a sequence of bits having a given rate. Therefore in order to design an optimal digital communication system, in the sense of minimizing distortion between the source and its reproduction at the receiver, it suffices to design an encoder-decoder pair that operate at rate below the channel capacity and achieves distortion predicted by the distortion rate function at that rate. In deriving such a result, it is assumed that encoder and decoder operate on infinitely long sequences. Also it is implicitly assumed that the source coder and the channel coder can be designed separately and then can be concatenated without any loss in the performance. This result is only asymptotic and not quite valid in practical situation where we deal only with finite lengths in which case the system resulting after concatenation may not be optimal. Under these conditions, the correct approach would be to design the source and channel coders jointly.

In the second part of the dissertation, we consider a joint source-channel coding problems for a continuous alphabet source and a noisy discrete memoryless channel. In the past, combined source-channel coding has been studied in various quantization contexts. Design algorithms for an optimum scalar quantizer operating over a noisy channel were developed in [25], [26]. These ideas were extended to LBG-VQ in [27] and then to tree-structured VQ (TSVQ) and multi-stage VQ (MSVQ) in [28]. Furthermore, necessary conditions of optimality for trellis encod-

ing systems have been developed in [29] and a design algorithm based on these necessary conditions is described in [30]. An FSVQ can be looked upon as a trellis encoding system with unit search depth and a general next-state function (as opposed to the one imposed by the shift-register implementation). The problem of combined source-channel coding for FSVQ can be developed along the lines of the system in [29]. However, due to the use of a general next-state function, the solution of [29] for the trellis encoding system cannot be used as such. As a consequence, we make simplifying assumptions about the FSVQ structure and describe two noisy channel FSVQ systems (NC-FSVQ1 and NC-FSVQ2). In NC-FSVQ1, we assume that the "protected" encoder state is transmitted periodically to the decoder, while in NC-FSVQ2, we try to modify the FSVQ structure in such a manner that without loosing too much in terms of performance (as compared to FSVQ of [18] in noiseless case), the theory of [29] can be used after some modifications. One important application that we will consider for noisy channel case will be that of quantizing the speech LSP parameters. The speech LSP parameters represent the short-time spectrum of speech and are mathematically equivalent to other linear predictive coding representations. Currently, there is a growing interest in encoding speech LSP parameters [31]-[33] used in vocoders [34] and hybrid speech coders [35]-[38]. The LSP parameters exhibit high intraframe and interframe correlations and therefore are good candidates for FSVQ type systems. Next, we briefly describe the speech LSP parameters.

1.2.1 LSP Parameters

For a given order p, the linear predictive analysis of speech results in an all-pole filter $H(z) = 1/A_p(z)$, where

$$A_p(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_p z^{-p}. \tag{1.6}$$

The parameters $\{a_i\}_{i=1,2,\ldots,p}$ are commomly referred to as LPC coefficients. It can be easily verified that the polynomial $A_j(z)$, associated with a j^{th}-order LPC analysis, satisfies the following recurrence relationship [39]

$$A_j(z) = A_{j-1}(z) - k_j A_{j-1}(z^{-1}), j = 1, 2, \ldots, p. \tag{1.7}$$

In (1.7), consider two extreme artificial boundary conditions: $k_{p+1} = 1$ and $k_{p+1} = -1$ corresponding, respectively, to the complete closure and a complete opening at the glottis in the acoustic tube model [39]. Under these conditions, the polynomial $A_{p+1}(z)$ can be expressed as

$$P(z) = (1 - z^{-1}) \prod_{i=2,4,\ldots,p} (1 - 2z^{-1}\cos\omega_i + z^{-2}), \tag{1.8}$$

for $k_{p+1} = 1$ and

$$Q(z) = (1 + z^{-1}) \prod_{i=1,3,\ldots,p-1} (1 - 2z^{-1}\cos\omega_i + z^{-2}), \tag{1.9}$$

for $k_{p+1} = -1$, where it is assumed that p is an even integer.

It is clear from (1.8) and (1.9) that $e^{j\omega_i}$, $i = 1, 2, \ldots, p$, are the roots of polynomial $P(z)$ and $Q(z)$. The parameters $\{\omega_i\}_{i=1,2,\ldots,p}$ are defined as the *line spectrum pair* (LSP) parameters. Note that $\omega_0 = 0$ and $\omega_{p+1} = \pi$ are the fixed roots of $P(z)$ and $Q(z)$, respectively. The LSP parameters can be interpreted as the resonant frequencies of the vocal tract under two extreme artificial boundary conditions at the glottis [34]. It has been demonstrated in [34] that LSP parameters have certain distinct properties that make them attractive for quantization in comparison with other equivalent LPC representations. In all the experiments involving LSP parameters, the final performance measure is considered to be the average spectral distortion (SD) given by

$$\text{SD} = \frac{1}{T}\sum_{n=1}^{T}[\int_{-\pi}^{\pi}(10\log S_n(\omega) - 10\log \hat{S}_n(\omega))^2 \frac{d\omega}{2\pi}]^{\frac{1}{2}}(dB). \tag{1.10}$$

Here $S_n(\omega)$ and $\hat{S}_n(\omega)$ are the original and quantized spectra, respectively, associated with the n^{th} frame and T is the number of frames. The average spectral distortion is established as a standard for measuring the performance of quantization schemes for LSP parameters [34]. In fact, the quantization of LSP parameters is considered to be transparent if the SD is below 1.0 dB and the percentage of frames with spectral distortion greater than 2.0 dB (called outliers) is less than 2% with no frames having spectral distortion greater than 4.0 dB.

1.3 Outline of the Thesis

The thesis consists of 5 chapters. In chapter 2, FSVQ of [18] is generalized to cases where the state quantizers can be any kind of VQ (as opposed to just LBG-VQ in [18]) and then variable-rate versions of various FSVQs are described, designed and evaluated for two types of sources namely a composite Gauss-Markov source and sampled speech waveform. In Chapter 3, FSVQs described in Chapter 2 are extended to 2-dimensional case (2-D) and then two systems based on 2-D FSVQs are described and used to encode *still* images. We assume a noiseless channel throughout Chapters 2 and 3. However, in Chapter 4 we consider noisy channels and provide descriptions and design algorithms for noisy channel FSVQs; we first justify the need to redesign FSVQ under noisy channel conditions and then attempt to provide necessary conditions of optimality for such systems. Only fixed rate FSVQ systems are considered and simulation results are provided for the Gauss-Markov source and speech LSP parameters. Finally Chapter 5 contains conclusions and suggestions for future research.

Chapter 2

Variable-Rate Finite-State Vector Quantizer

2.1 Introduction

In [18], the FSVQ was introduced and used to encode two sources with memory, namely the 1^{st}-order Gauss-Markov source and sampled speech waveform; performance improvements were observed as compared to the ordinary memoryless VQ (LBG-VQ). Then in [19], a technique based on adaptive stochastic automata theory was introduced that led to an improved design algorithm for FSVQ and an application of FSVQ was made to voice coding. Several schemes based on FSVQ have also been reported in the image coding literature. In [20] and [21], FSVQ was used to encode still images where the state was used to exploit the correlation in the spatial domain; over 50% saving in bit rate is achieved over LBG-VQ. In [22] and [23], FSVQ was used in coding image sequences where the state is defined to exploit temporal correlation; again a saving of over 50% is achieved over intraframe LBG-VQ.

As mentioned in Chapter 1, in all the FSVQ systems considered in literature so far, the state codebooks are all assumed to have the same cardinality and hence the same bit rate. This, in a loose sense, implies that all the states are treated with approximately equal degree of fidelity. Such a restriction limits the performance of

the FSVQ which has the potential of doing better. In this chapter, we have relaxed this assumption and have considered an FSVQ scheme with different codebook sizes for different states while constraining the average bit rate. This modification has resulted in performance improvements (in some cases substantial) in terms of signal-to-noise ratio. We have also considered the possibility of using structured VQs such as the tree-searched VQ (TSVQ) [8] and optimally pruned unbalanced TSVQ (UTSVQ) [24] as the state quantizers. In fact, the scheme using UTSVQ as state quantizers and with the flexibility of having variable bit rate assignment among the states performs the best amongst all the schemes considered in this thesis (under noiseless channel conditions), as will be shown by the simulation results.

In order to facilitate the presentation and avoid using long names for different kinds of systems described in this chapter, we will use appropriately defined acronyms. Furthermore, as an aid to remember the correspondence between different acronyms and their respective systems, we provide the notation tree of Fig. 2.1. In this figure, FSVQ refers jointly to fixed-rate as well as variable-rate finite-state vector quantization. Furthermore, fixed-rate FSVQ with LBG-VQ as state quantizers is called FS-LBGVQ and when TSVQs are used as state quantizers then the fixed-rate FSVQ system is referred to as FS-TSVQ. Similarly, variable-rate FSVQ with LBG-VQ is called VFS-LBGVQ and the one with TSVQ as state quantizers is named VFS-TSVQ; finally, we use VFS-UTSVQ to denote variable-rate FSVQ with UTSVQs as state quantizers.

In this chapter, we describe the 1-dimensional (1-D) versions of various fixed-rate and variable-rate FSVQs mentioned above and use them to encode sampled speech and a synthetic source described in Section 2.4.

The rest of the chapter is organized as follows. In Section 2.2, we provide the definition and design algorithm for fixed-rate FSVQ with both structured and

unstructured codebooks. The description and design algorithm for variable-rate FSVQ is provided in Section 2.3, while the simulation results of the various FSVQ based systems on 1-D sources like sampled speech waveform and a switched Gauss-Markov source are given in Section 2.4. Finally, a summary and conclusions are given in Section 2.5

2.2 Fixed-Rate Finite-State Vector Quantization

In this section, we briefly provide the description and design algorithm of fixed-rate FSVQ with both structured and unstructured state codebooks. We first provide the definition and design algorithm for fixed-rate FSVQ with unstructured codebooks (denoted by FS-LBGVQ). Following that, we will consider fixed-rate FSVQ with tree-structured codebooks (denoted by FS-TSVQ).

2.2.1 Definition of FS-LBGVQ

An L-dimensional K-state FS-LBGVQ [18] is specified by a state space $\mathcal{S} = \{1, 2, \ldots, K\}$, an initial state s_0 and three mappings:

(1) $\alpha : \mathbb{R}^L \times \mathcal{S} \rightarrow \mathcal{N}$: finite-state encoder,

(2) $\beta : \mathcal{N} \times \mathcal{S} \rightarrow \hat{\mathcal{A}}$: finite-state decoder,

(3) $f : \mathcal{N} \times \mathcal{S} \rightarrow \mathcal{S}$: next state function.

Here, $\mathcal{N} \triangleq \{1, 2, \ldots, N\}$ is the finite channel alphabet of size N and $\hat{\mathcal{A}}$ is the reproduction space.

Let $\{\mathbf{x}_n\}_{n=0}^{\infty}$ denote the input vector sequence, where $\mathbf{x}_n \in \mathbb{R}^L$. Similarly let $\{u_n\}_{n=0}^{\infty}$, $\{s_n\}_{n=0}^{\infty}$ and $\{\hat{\mathbf{x}}_n\}_{n=0}^{\infty}$ denote the channel symbol sequence, state sequence and reproduction vector sequence, respectively. With initial state s_0, the input process determines the sequence of channel symbols, reproduction vectors and

states according to:

$$u_n = \alpha(\mathbf{x}_n, s_n), \quad n = 0, 1, \ldots, \tag{2.1.a}$$

$$\hat{\mathbf{x}}_n = \beta(u_n, s_n), \quad n = 0, 1, \ldots, \tag{2.1.b}$$

$$s_{n+1} = f(u_n, s_n), \quad n = 0, 1, \ldots. \tag{2.1.c}$$

Note that the next state depends only on the present state and the output channel symbol, and therefore, given the initial state and correct channel symbol sequence, the decoder can track the state sequence. The collection $\mathcal{C}_k \triangleq \{\beta(u, k), u \in \mathcal{N}\}$ is the codebook associated with state k; obviously, $\hat{\mathcal{A}} = \bigcup_{k=1}^{K} \mathcal{C}_k$. For a given state space $\mathcal{S}$ and a channel alphabet $\mathcal{N}$, the mapping β can be stored as a look-up table for a given FS-LBGVQ. The rate of an FS-LBGVQ is given by $R = \log_2 N$, bits/vector.

The encoder mapping is specified in terms of a distortion function that is used to measure the performance of the FS-LBGVQ. The distortion measure $d : \mathbb{R}^L \times \hat{\mathcal{A}} \rightarrow [0, \infty)$ assigns a non-negative cost $d(\mathbf{x}, \hat{\mathbf{x}})$ to reproducing the input vector $\mathbf{x}$ as $\hat{\mathbf{x}}$. Then the encoder is specified by the minimum distortion rule [18]

$$\alpha(\mathbf{x}, k) = \arg\min_{u \in \mathcal{N}} d(\mathbf{x}, \beta(u, k)), \forall k \in \mathcal{S}. \tag{2.2}$$

An FS-LBGVQ can be interpreted as a set of K LBG-VQs (one LBG-VQ associated with each state), each of codebook size N [4], [17]. The current input vector is vector-quantized using the LBG-VQ associated with the current state of the system; the current state and channel symbol determine the next state.

2.2.2 Design Algorithm of FS-LBGVQ

Given a training sequence $\{\mathbf{x}_n, n = 0, 1, \ldots\}$ and a distortion measure, the design algorithm for an L-dimensional, K-state FS-LBGVQ of rate $R = \log_2 N$, bits/vector consists of designing: (a) the state codebooks $\mathcal{C}_k$ for FS-LBGVQ, each

of size N, and (b) the next-state function $f(u,k)$, $u \in \mathcal{N}$, $k \in \mathcal{S}$. Following [18], the design algorithm can be described in four steps as described below.

1. Design an LBG-VQ [4] with K codevectors for the given training sequence. We refer to this VQ as the state-label VQ, $\mathcal{C} = \{\mathbf{c}(k), k \in \mathcal{S}\}$.

2. For each state k of the FS-LBGVQ, design an initial reproduction codebook $\mathcal{C}_k = \{\beta(u,k), u \in \mathcal{N}\}$ using the LBG algorithm [4] on the sub-training sequence composed of all successors to vectors which are represented by k if quantized by the state-label VQ, i.e., the subsequence $\{\mathbf{x}_n : k = \arg\min_{s\in\mathcal{S}} d(\mathbf{x}_{n-1}, \mathbf{c}(s))\}$. Thus each codebook $\mathcal{C}_k$ is designed to be good for vectors which will occur next if the FS-LBGVQ is currently in the ideal state k.

3. The ideal state k in step (2) depends on the input vector and therefore the decoder at the receiver side will not be able to track the ideal state sequence. In order to enable the decoder to track the state sequence (without transmitting any overhead information), we choose the next state as the label which best matches the *reproduction* of the current input vector rather than the current input vector itself. Thus given the state labels $\mathbf{c}(k)$ and the decoder β designed in step (2), we define a next-state function f by

$$f(u,k) = \arg\min_{s\in\mathcal{S}} d(\beta(u,k), \mathbf{c}(s)),\ k \in \mathcal{S}, u \in \mathcal{N}. \tag{2.3}$$

4. Attempt to improve the state codebooks $\{\mathcal{C}_k, k \in \mathcal{S}\}$ of the FS-LBGVQ by encoding the training sequence using the next-state function obtained in step (3) and updating each codevector by replacing it by the centroid of the cell associated with that codevector. Also update the state-label VQ $\mathcal{C}$ similarly [18].

In most cases further improvements are possible by iterating steps (3) and (4). The algorithm described above does not necessarily converge and in fact it does not even guarantee improved performance at each step of iteration. However, the FS-LBGVQs designed with this algorithm exhibit substantial gain over LBG-VQs in both waveform coding and vocoding applications [18], [19].

2.2.3 Fixed-Rate FSVQ Based on Tree-Structured Vector Quantizer

A finite-state tree-structured vector quantizer (FS-TSVQ) is specified in a manner very similar to FS-LBGVQ. Associated with each state, we now have a TSVQ rather than an LBG-VQ. The encoding is accordingly done in a tree-structured manner. In particular, the encoder mapping $\alpha : \mathbb{R}^L \times \mathcal{S} \rightarrow \mathcal{N}$ for FS-TSVQ differs from that of the FS-LBGVQ; rather than computing the index of the minimum-distortion codevector in the state codebook, we now encode the input vector using the state TSVQ and the output channel symbol is the index of the codeword resulting from the TSVQ encoding. The main advantage of this scheme is the complexity reduction in the encoding obtained due to the structured nature of the state codebooks without much loss in performance. The design algorithm for FS-TSVQ is very similar to that of FS-LBGVQ given in Subsection 2.2.2. The only difference lies in the encoding procedure as mentioned above and in step (2) of the algorithm, where we now design a TSVQ for each state instead of an LBG-VQ. As in the case of FS-LBGVQ, the design algorithm for FS-TSVQ does not necessarily converge and there is no guaranteed performance improvement at each step of iteration.

In the FS-LBGVQ design algorithm presented by Foster *et al* [18], all the state codebooks are assumed to have the same cardinality and in order to design a rate R bits/vector FS-LBGVQ, the cardinality of each state codebook is assumed to be 2^R.

In the next section, we drop this assumption and present a modified FS-LBGVQ system (and also a modified FS-TSVQ system), in which the state codebook sizes are allowed to vary from state to state; this leads to performance gains (in some cases substantial). We also consider a second variation of the FS-LBGVQ system in which UTSVQs obtained by optimal pruning of complete TSVQs [24] are used as state VQs and state rates are not constrained to be the same for all the states.

2.3 Variable-Rate Finite-State Vector Quantization

So far, the state VQs were assumed to have the same cardinality (and hence the same bit rate). Roughly speaking, this assumption implies that the source vectors are encoded with more-or-less the same degree of fidelity regardless of the state. This assumption may be unnecessarily restrictive in certain applications where some types of source vectors should be quantized more finely than some others (e.g., in speech coding silence periods can be quantized quite coarsely). In what follows we relax this assumption and let the rates of state VQs vary from state to state subject to a constraint on the average encoding rate.

2.3.1 Definition of Variable-Rate FSVQ

A variable-rate FSVQ is specified by a state space $\mathcal{S} = \{1, 2, \ldots, K\}$, an initial state s_0 and three mappings as follows:

(1) $\alpha : \mathbb{R}^L \times \mathcal{S} \rightarrow \mathcal{N}(\mathcal{S})$: finite-state encoder,

(2) $\beta : \mathcal{N}(\mathcal{S}) \times \mathcal{S} \rightarrow \hat{\mathcal{A}}$: finite-state decoder,

(3) $f : \mathcal{N}(\mathcal{S}) \times \mathcal{S} \rightarrow \mathcal{S}$: next state function.

Here the channel alphabet depends on the state ($\mathcal{N}(k)$ for state k) and, in general, is different for each state. Hence the system becomes a variable-rate system. Accordingly, the rate of the system is defined by $R = \sum_{k=1}^{K} P_k b_k$, bits/vector, where

b_k is the average rate of the quantizer associated with state k and P_k is the probability of occurrence of state k. The operation of variable-rate FSVQ is the same as that of fixed-rate FSVQ. It can again be interpreted as a set of K state quantizers, one associated with each state and the bit rate of the quantizer associated with state k is b_k. The current input vector is encoded using the quantizer associated with the current state of the system; the current state and the channel symbol determine the next state. In the sequel, we will refer to the variable-rate versions of FS-LBGVQ and FS-TSVQ by VFS-LBGVQ and VFS-TSVQ, respectively.

2.3.2 Design of VFS-LBGVQ

A VFS-LBGVQ differs from an FS-LBGVQ due to the fact that now the bit rates associated with different state codebooks are not constrained to be the same. As a consequence, state codebook sizes become an additional set of variables in the design stage of VFS-LBGVQ. Once the codebook sizes associated with states are determined, the design algorithm for VFS-LBGVQ can be described along the lines of the design algorithm for FS-LBGVQ. The bit-rate assignment (codebook size determination) is done using the concept of optimal pruning of TSVQ [24] described next. Following that, we describe the bit assignment algorithm and the complete design algorithm for VFS-LBGVQ.

Optimal Pruning of a TSVQ

Consider a complete binary TSVQ of rate l bits/vector. Corresponding to this TSVQ, there is a complete binary tree of depth l with 2^l leaves. Associated with each interior node (not including the root node) and leaf of the tree, there is a codevector (reproduction level), a probability and a conditional expected distortion. By pruning off various branches of the tree, a variable-rate TSVQ or an unbalanced TSVQ (UTSVQ) is obtained. The codebook of the UTSVQ is the set

of the codevectors associated with the leaves of the pruned tree. The quantizer's average rate is the sum, over all the leaves, of the leaf probability times the length from the root to the leaf. The quantizer's average distortion is the sum, over all the leaves, of the leaf probability times the conditional expected distortion associated with the leaf.

Now suppose $\mathcal{T}$ is a large tree corresponding to a complete (completeness is not mandatory) TSVQ, then every pruned subtree $\mathcal{P}$ of $\mathcal{T}$ ($\mathcal{P} \preceq \mathcal{T}$) defines a UTSVQ with average rate $\ell(\mathcal{P})$ and average distortion $\delta(\mathcal{P})$. The operational distortion-rate performance defined by

$$\hat{D}(R) = \min_{\mathcal{P} \preceq \mathcal{T}} \{\delta(\mathcal{P}) | \ell(\mathcal{P}) \leq R\} \tag{2.4}$$

specifies the optimal trade-off between rate and distortion over all pruned subtrees of $\mathcal{T}$. A reinterpreted version of an algorithm developed in the context of classification and regression trees [40] is presented in [24] which traces out the convex-hull of the operational distortion-rate performance. The algorithm given in [24] is quite general and if δ is any monotone decreasing real-valued function defined on trees (i.e., if $\mathcal{P}_1 \preceq \mathcal{P}_2 \preceq \mathcal{T}$, then $\delta(\mathcal{P}_1) \geq \delta(\mathcal{P}_2)$) and ℓ is any monotone increasing real-valued function defined on trees, then the algorithm gives the optimal trade-off between ℓ and δ over all pruned subtrees of $\mathcal{T}$.

Bit Rate Assignment Algorithm

Suppose we are given K sets of codebooks $\{\mathcal{C}_k^i, i = 1, 2, \ldots, M_k\}$, $k \in \mathcal{S}$, one set associated with each state. Let the rate (in bits/vector) and average distortion associated with $\mathcal{C}_k^i$ be given by $R_{k,i}$ and $D_{k,R_{k,i}}$, respectively. Also let $R_{k,1} < R_{k,2} < \ldots < R_{k,M_k}$, $\forall k \in \mathcal{S}$. Then, $D_{k,R_{k,1}} \geq D_{k,R_{k,2}} \geq \ldots \geq D_{k,R_{k,M_k}}$, $\forall k \in \mathcal{S}$. Given the set of codebooks $\{\mathcal{C}_k^i, i = 1, 2, \ldots, M_k\}, k \in \mathcal{S}$, we want to choose a codebook of bit rate b_k^* from each set as the state codebook of the VFS-LBGVQ, i.e., determine

the bit rate assignment map $(b_1^*, b_2^*, \ldots, b_k^*)$ that minimizes the average distortion given by

$$D = \sum_{k=1}^{K} P_k D_{k,b_k}, \tag{2.5.a}$$

subject to

$$\sum_{k=1}^{K} P_k b_k \leq b_{avg}, \tag{2.5.b}$$

and

$$b_k \in \{R_{k,1}, R_{k,2}, \ldots, R_{k,M_k}\}, \tag{2.5.c}$$

where b_{avg} is the desired average rate in bits/vector.

The above bit assignment problem can be solved using the idea of optimal pruning of a TSVQ as follows: We first construct a tree $\mathcal{T}$ (see Fig. 2.2) the root node of which has K children, one per state, and the subtree rooted at each child k is a unary tree of length M_k . Thus we have K branches, each associated with a state, coming out of the root node of the tree. Let each node of the branch associated with state k correspond to a codebook from $\{\mathcal{C}_k^i,\ i = 1, 2, \ldots, M_k\}$ and hence to a rate-distortion pair; the node closest to the root of the tree has rate 1 bit/vector (and distortion $D_{k,1}$) and in increasing order the node farthest from the root node has rate R_{k,M_k} (and distortion $D_{k,R_{k,M_k}}$).

Let $\mathcal{P}$ be a pruned subtree of $\mathcal{T}$ with the branch associated with state k of length l_k. Corresponding to this pruned tree $\mathcal{P}$, we construct a VFS-LBGVQ system with $\mathcal{C}_k^{l_k}$ as the state codebook associated with state k, $\forall k \in \mathcal{S}$. We assume for the moment that the next-state function is given to us; the problem of determining the next-state function is considered in the next subsection. Then the rate of the VFS-LBGVQ associated with $\mathcal{P}$ is given by

$$\ell(\mathcal{P}) = \sum_{k=1}^{K} P_k R_{k,l_k}, \tag{2.6}$$

and the average distortion is given by

$$\delta(\mathcal{P}) = \sum_{k=1}^{K} P_k D_{k,R_{k,l_k}}. \tag{2.7}$$

The optimal pruning algorithm of [24], when applied to the tree $\mathcal{T}$ constructed above gives the optimal pruned subtree $\mathcal{P}^*$ and hence the bit rate assignment map $(b_1^*, b_2^*, \ldots, b_K^*)$ that minimizes $\delta(\mathcal{P})$ subject to $\ell(\mathcal{P}) \leq b_{avg}$ over all $\mathcal{P} \preceq \mathcal{T}$.

Design Algorithm for VFS-LBGVQ

Let D_{k,b_k} denote the average distortion incurred in state k when the rate of the quantizer associated with state k is b_k bits/vector. Then we wish to minimize the average distortion given by

$$D = \sum_{k=1}^{K} P_k D_{k,b_k}, \tag{2.8}$$

subject to a constraint on the average rate described by

$$\sum_{k=1}^{K} P_k b_k \leq b_{avg}, \tag{2.9}$$

by appropriately designing the bit assignment map $(b_1^*, b_2^*, \ldots, b_K^*)$, the state codebooks $\{\mathcal{C}_k, k \in \mathcal{S}\}$ for VFS-LBGVQ and the next state function. An additional constraint implicitly assumed is that the rate in bits/vector associated with each state quantizer is constrained to be an integer. The design algorithm consists of the following steps:

1. For the given training sequence, design the state-label VQ, $\mathcal{C} = \{\mathbf{c}(k), k \in \mathcal{S}\}$, using the LBG algorithm [4].

2. For each state k, construct the subtraining sequence consisting of the subsequence $\{\mathbf{x}_n : k = \arg\min_{s \in \mathcal{S}} d(\mathbf{x}_{n-1}, \mathbf{c}(s))\}$. Then, for each state k, design LBG-VQs of rates $1, 2, \ldots, b_{max,k}$ [1] bits/vector. We denote the set of LBG-

[1] $b_{max,k}$ is determined based on the size of the subtraining sequence associated with the state k. It is determined such that each quantizer bin is richly populated so that the codevector associated with that bin is a meaningful representative of the training vectors assigned to that bin.

VQs by $\{\mathcal{C}_k^i, k \in \mathcal{S}, i = 1, 2, \ldots, M_k\}$, where $M_k = b_{max,k}$.

3. Find the optimum bit assignment map $(b_1^*, b_2^*, \ldots, b_K^*)$ using the algorithm described in in the previous subsection. The state codebook used for state k will be $\mathcal{C}_k^{b_k^*} = \{\beta^{b_k^*}(u,k), u \in \{1, 2, \ldots, 2^{b_k^*}\}\}$ for VFS-LBGVQ, where $\beta^{b_k^*}$ is the finite state decoder, associated with state k, with channel alphabet $\{1, 2, \ldots, 2^{b_k^*}\}$.

4. As in the case of FS-LBGVQ, the next-state function f is defined as

$$f(u,k) = \arg\min_{s \in \mathcal{S}} d(\beta^{b_k^*}(u,k), \mathbf{c}(s)),\ k \in \mathcal{S}, u \in \{1, 2, \ldots, 2^{b_k^*}\}. \quad (2.10)$$

5. Encode the entire training sequence using the next-state function f and the state codebooks $\{\mathcal{C}_k^{b_k^*}, k \in \mathcal{S}\}$ for VFS-LBGVQ. After encoding, update the state-label VQ by replacing each $\mathbf{c}(k)$ by the conditional centroid of the cell associated with it. As a result of encoding, each state k has a subtraining sequence associated with it given by the subsequence $\{\mathbf{x}_n : k = f(\alpha(\mathbf{x}_{n-1}, s_{n-1}), s_{n-1})\}$. It differs from the subsequences of step (2) due to the introduction of the next-state function in the encoding process.

6. For all $k \in \mathcal{S}$, update the codebooks $\{\mathcal{C}_k^i, i \in \{1, 2, \ldots, b_{max,k}\}\}$ by encoding the training sequence associated with state k and replacing each reproduction level of $\mathcal{C}_k^i$ by the conditional centroid of the cell associated with the codevector. Also update state-label VQ $\mathcal{C}$ similarly [18]. Then repeat steps (3), (4), (5) and (6) for some predetermined number of iterations or until convergence. Then among all quantizers obtained select the one with the best rate-distortion performance.

As in the case of FS-LBGVQ, the design algorithm does not necessarily converge and it does not even guarantee improvement at each step of iteration. However,

the system obtained using this algorithm performs better than FS-LBGVQ as will be shown by the results in Section 2.4.

2.3.3 Variable-Rate FSVQ Based on TSVQ

As in the fixed-rate case, we can describe a variable-rate FSVQ based on TSVQ (VFS-TSVQ) as a VFS-LBGVQ in which each state LBG-VQ is replaced by a TSVQ. Again the advantage of VFS-TSVQ over VFS-LBGVQ is the computational complexity reduction without much performance loss. The design algorithm for VFS-TSVQ is the same as described in Subsection 2.3.2 except that step (6) in the algorithm is replaced by the following step (6′):

6′. For all $k \in \mathcal{S}$, redesign the TSVQ codebooks $\{\mathcal{C}_k^i, i \in \{1, 2, \ldots, b_{max,k}\}\}$ by using the training sequence associated with state k. Also update the state-label VQ $\mathcal{C}$. Then repeat steps (3), (4), (5) and (6′) for some predetermined number of iterations or until convergence. Then among all quantizers obtained select the one with the best rate-distortion performance.

In the step 6 of the design algorithm of VFS-LBGVQ, the state codebooks are not redesigned; they are just updated (each codevector is replaced by the centroid of its respective encoding cell) once. In case of designing VFS-TSVQ, however, state codebooks are redesigned in step 6′, which is practical due to the lower computational complexity of designing TSVQ as compared to designing LBG-VQ of the same rate.

Variable-Rate FSVQ Based on UTSVQs

In another variation of the FS-LBGVQ scheme, we consider a system in which the state quantizers are optimally pruned UTSVQs obtained using the algorithm in [24]. We also have the flexibility of having different rates for different states. The main motivation behind using such a scheme was the superior performance of the

optimally pruned UTSVQ over LBG-VQ along with the additional advantage of fast encoding due to the tree-searched method. In this scheme, even for a given state, the quantizer is a variable-rate encoder; for VFS-LBGVQ (VFS-TSVQ) the rate varies between states but is fixed within each state. We refer to the new scheme as VFS-UTSVQ. The system is formally described in the same way as VFS-TSVQ. The design algorithm is also similar to that of VFS-TSVQ with step (2) modified in the following way:

2'. For each state k, we design a complete TSVQ of rate $\hat{b}_{max,k}$, where $\hat{b}_{max,k}$ is determined in the same way as $b_{max,k}$ is determined in the design of VFS-TSVQ. Then, using the optimal pruning algorithm of [24] on each of the complete state TSVQs, we obtain K sets of collection of optimally pruned UTSVQs. The rate of the optimally pruned UTSVQs associated with state k varies from 1 bit/vector to $\hat{b}_{max,k}$ bits/vector and takes finitely many values which are not necessarily integers; the fact that rates are not constrained to be integers as in VFS-LBGVQ (VFS-TSVQ) leads to an additional improvement factor. Then we apply the bit assignment algorithm on the collection of K sets of optimally pruned UTSVQs to obtain the optimal bit rate assignment map $(b_1^*, b_2^*, \ldots, b_K^*)$.

The remaining steps of the algorithm are identical to the design algorithm of VFS-TSVQ. Again, the algorithm is suboptimal but the final system obtained using this algorithm gives performance gains over all other schemes considered in this chapter.

2.4 Simulation Results for 1-D Sources

We performed extensive simulations to compare the variable-rate FSVQ systems described in this chapter with the FS-LBGVQ scheme described in [18]. The

performance comparisons were made for two kinds of 1-D sources: (i) a synthetic switched Gauss-Markov (G-M) source and (ii) sampled speech waveform. The algorithms described in the previous sections were used to design FS-LBGVQ, FS-TSVQ, VFS-LBGVQ, VFS-TSVQ and VFS-UTSVQ. Since the algorithms did not necessarily converge, we carried out 20 iterations for each scheme and chose the best case. The performance measure used is the signal-to-quantization-noise ratio (SQNR) in dB. All results are obtained for vector dimension $L = 8$ and number of states $K = 8$, 16 and 32. In the sequel, we denote by b the average bit rate per sample.

2.4.1 Results on the Synthetic Source

For a simple stationary source such as the 1^{st}-order G-M source with correlation coefficient 0.9, we found experimentally that fixed-rate FSVQ achieved a gain of about 1 dB over LBG-VQ in terms of SQNR. However, the performance of all variable-rate FSVQ schemes were found to be quite close to that of fixed-rate FSVQ. For this simple source, it is observed that the performance of FS-LBGVQ is within 1.0-1.6 dB of the rate-distortion function for the rates considered in this chapter and therefore, we do not expect the variable-rate versions of FSVQ to achieve any significant gain over FS-LBGVQ. Similar observations were made in a slightly different context by [24] for this G-M source. Therefore, for benchmarking purposes, we have considered a synthetic, more complex switched source in which at the beginning of each switching period a hidden mechanism chooses, according to a Markovian transition, between two 1^{st}-order G-M sources with different variances; in [24], a similar switched source (however, with memoryless subsources) was considered. For the example given here, the switch transition probability matrix is chosen as

$$\begin{pmatrix} p_{0,0} & p_{0,1} \\ p_{1,0} & p_{1,1} \end{pmatrix} = \begin{pmatrix} 0.98 & 0.02 \\ 0.02 & 0.98 \end{pmatrix}$$

where $p_{i,j}$ is the probability of switching from subsource i to j, $\forall\ i,j = 0,1$. For both G-M subsources, the correlation coefficients were taken to be 0.9; the variances of the two sources were chosen to be different by a factor of 1000. The training sequence consisted of 150,000 vectors. For testing, a different sequence of 150,000 vectors was used.

Table 2.1 shows the performance of FS-LBGVQ. The LBG-VQ performance results are also included in the table for comparison. For $K = 8$, FS-LBGVQ performs better than LBG-VQ by about 0.6-0.9 dB and the gain increases with K. Table 2.2 summarizes the performance of FS-TSVQ and TSVQ. As compared to FS-LBGVQ, FS-TSVQ performs slightly worse due to the use of TSVQ in place of LBG-VQ.

Tables 2.3 and 2.4 summarize the performance of VFS-LBGVQ and VFS-TSVQ, respectively. In these and subsequent tables, the numbers in parentheses indicate the achieved bit rate while b is the design bit rate. The performance improvements of the VFS-LBGVQ (VFS-TSVQ) over the FS-LBGVQ (FS-TSVQ) schemes are evident. Finally, Table 2.5 contains the performance results for VFS-UTSVQ, which performs the best among the five schemes. As compared to the second best scheme VFS-LBGVQ, it achieves about 0.15-0.5 dB gain in terms of SQNR. Also, for comparison purposes, we have included the results of UTSVQ [24]. As a result of using finite-state vector quantization, a gain of over 1.5 dB in terms of SQNR is achieved over UTSVQ.

2.4.2 Results on Sampled Speech

The training sequence used consisted of five minutes of speech sampled at 8 KHz and uttered by five male and three female speakers.

The performance of FS-LBGVQ is summarized in Table 2.6. Also for comparison, we include the performance results of LBG-VQ. As observed in [18], the

FS-LBGVQ scheme outperforms the LBG-VQ by over 2 dB and the gain increases with the number of states. Table 2.7 summarizes the performance of FS-TSVQ. In this case, as compared to FS-LBGVQ, a slight degradation of performance can be observed. In general, however, the trend is similar to that of FS-LBGVQ. In most cases, FS-TSVQ yields a SQNR within 1.0 dB of FS-LBGVQ.

Table 2.8 summarizes the performance of VFS-LBGVQ for different values of b and K. Comparison of Tables 2.6 and 2.8 indicates that at the same bit rate, VFS-LBGVQ outperforms FS-LBGVQ, in general, by about 2.5 dB.

The performance of VFS-TSVQ is illustrated in Table 2.9. It should be noted that the difference between the performance of VFS-TSVQ and VFS-LBGVQ is smaller than the difference between FS-TSVQ and FS-LBGVQ. The reason resides in the limitation on the size of the largest LBG-VQ (2048 codevectors in codebook) needed for VFS-LBGVQs; this limitation is less severe for TSVQs.

Finally, we include the performance results of VFS-UTSVQ in Table 2.10. This system gives the best performance results among all the schemes considered in this chapter. For all values of K considered here, VFS-UTSVQ outperforms FS-LBGVQ by at least 3 dB and by as much as 4.25 dB for $b = 0.5$ bits/sample and higher. In order to visually compare the performance of the different systems, their rate-distortion performance on the training sequence are illustrated in Fig. 2.3, clearly demonstrating the superior performance of the variable-rate based schemes.

The performance of various FSVQ schemes on an out-of-training test sequence is summarized in Tables 2.11-2.15. The test sequence was 67 seconds of speech uttered by a male speaker and sampled at 8 KHz. These results are also depicted in Fig. 2.4. Study of these tables shows that even for out-of-training data, variable-rate FSVQs outperform their fixed-rate counterparts. It is important to note that in the variable-rate schemes, while the SQNR degrades for the out-of-training

sequence, the actual bit rate is also lower than the design rate b [2]. Comparison at the same bit rate (see Fig. 2.4) shows a gain of as much as 3 dB for VFS-UTSVQ and 2.4 dB for VFS-LBGVQ and VFS-TSVQ over the fixed-rate FSVQs. For the variable-rate schemes, a simple feedback mechanism can be used to sense the instantaneous output rate and adjust the bit rate accordingly by adjusting the encoder structure slightly; a feature that does not exist in a fixed-rate system. More interestingly, for the VFS-UTSVQ system, the bit rate adjustment can be achieved simply by adding/pruning branches of state codebooks. This is an additional important advantage over other variable-rate FSVQ systems (besides giving better performance at the same rate).

2.5 Summary and Conclusions

In this chapter, we have considered several variable-rate variations of the 1-D FS-LBGVQ scheme described in [18]. Design algorithm for various schemes are obtained by appropriately modifying the algorithm given in [18]. 1-D versions of FSVQ systems are used for encoding a composite Gauss-Markov source and sampled speech. It can be concluded from our results that, in general, the variable-rate versions of FSVQ lead to performance improvements which in certain cases are quite substantial. The best performance was achieved by the systems based on variable-rate FSVQ using UTSVQs as the state quantizers for both sources

Although the extension of fixed-rate FSVQ to variable-rate FSVQ leads to performance improvements, the potential problems associated with any variable-rate coding system like buffer overflow/underflow and channel error propagation will come into play. One possible method to deal with these problems is to use the idea of the recently developed structured vector quantizer [41], [42] to convert

[2]The actual rate achieved by the variable-rate FSVQ systems varies with the test sequence chosen. The use of a larger training sequence can reduce the variation in the actual rate.

the variable-rate FSVQ systems into a fixed-rate system without introducing a significant performance loss. By incorporating some delay into the system, the codebook search and encoding algorithm of [41] can be used to encode several vectors (say n) at a time such that the total number of bits used for the n vectors is fixed, but the number of bits used for each vector can be allowed to be variable. This modification will render the system a fixed-rate encoder while at the same time offers (to some extent) the advantages of the variable-rate systems. As a consequence, the effect of any channel error will remain confined to only n vectors; similarly the buffer overflow/underflow problem is eliminated since the encoder is a fixed-rate system now.

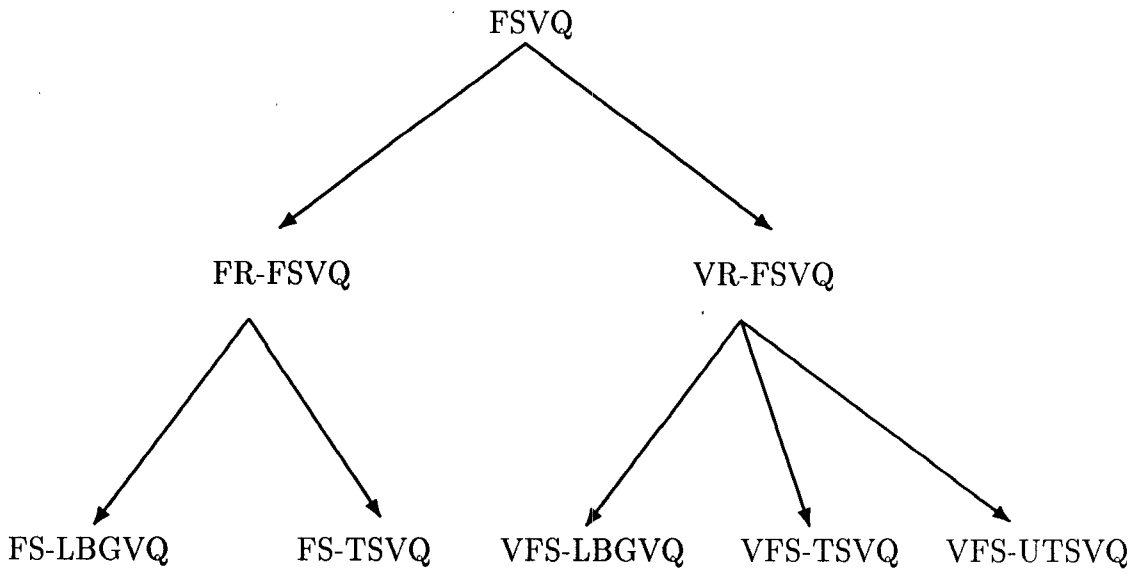


Figure 2.1: Notation tree.

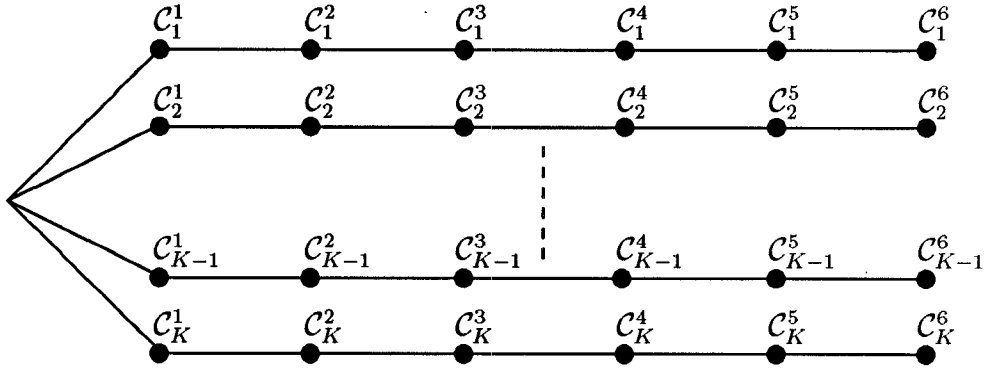


Figure 2.2: Tree used for bit assignment for VFS-LBGVQ; $M_k = 6$ for all states.

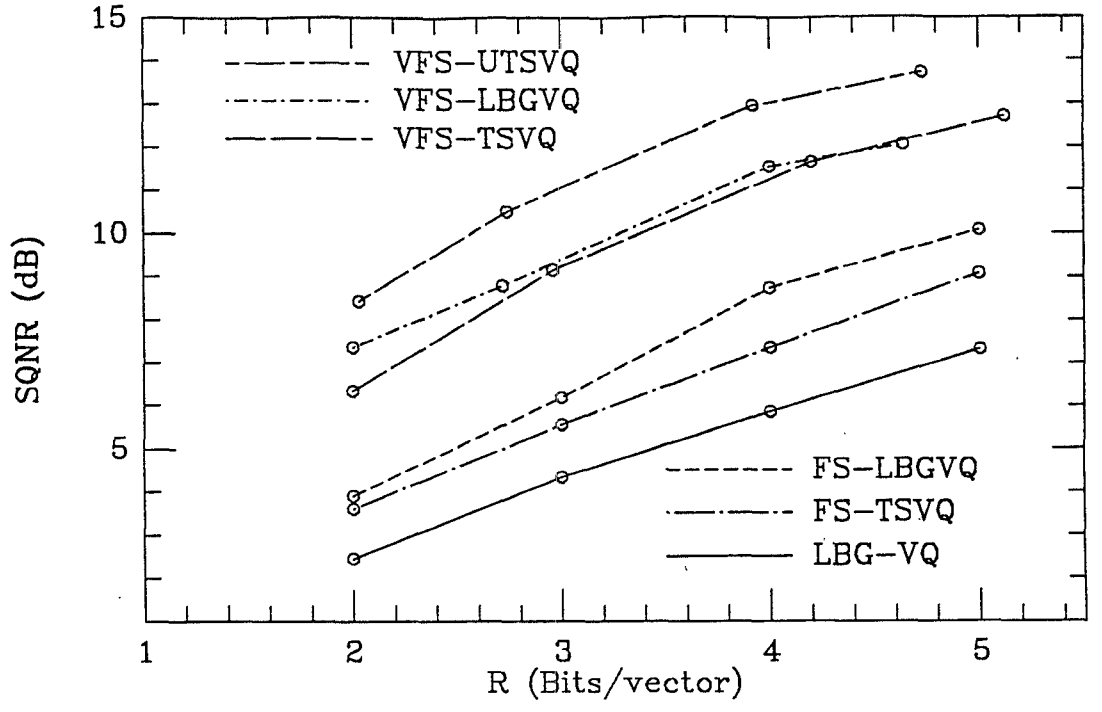


Figure 2.3: Performance of fixed-rate and variable-rate FSVQs on training sequence; $K = 32$.

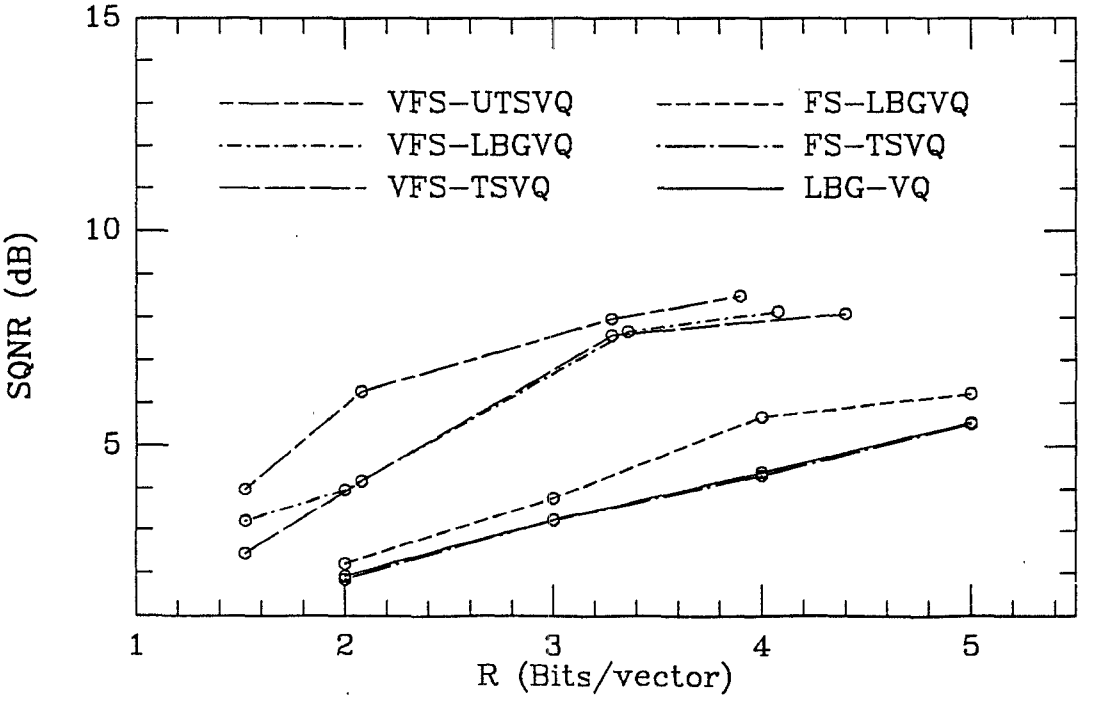


Figure 2.4: Performance of fixed-rate and variable-rate FSVQs on out-of-training sequence; $K = 32$.

	FS-LBGVQ		
	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR
8	7.36	8.47	9.43
16	7.41	8.57	9.55
32	7.45	8.62	9.62
LBG-VQ	6.50	7.74	8.83

Table 2.1: Performance of FS-LBGVQ and LBG-VQ at $b = 0.375$, 0.5 and 0.625 bits/sample on the Synthetic Source.

	FS-TSVQ		
	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR
8	7.00	8.11	9.06
16	7.15	8.23	9.19
32	7.23	8.29	9.23
TSVQ	5.87	6.98	8.38

Table 2.2: Performance of FS-TSVQ and TSVQ at $b = 0.375$, 0.5 and 0.625 bits/sample on the Synthetic Source.

	VFS-LBGVQ		
	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR
8	8.02 (0.38)	9.06 (0.49)	10.46 (0.63)
16	8.28 (0.38)	9.60 (0.50)	10.59 (0.63)
32	8.38 (0.39)	9.83 (0.50)	10.76 (0.63)

Table 2.3: Performance of VFS-LBGVQ at $b = 0.375$, 0.5 and 0.625 bits/sample on the Synthetic Source. Numbers in parentheses denote actual bit rate.

	VFS-TSVQ		
	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR
8	7.79 (0.375)	8.79 (0.48)	9.42 (0.59)
16	7.88 (0.37)	9.30 (0.50)	9.82 (0.56)
32	8.08 (0.375)	9.43 (0.50)	9.96 (0.58)

Table 2.4: Performance of VFS-TSVQ at $b = 0.375$, 0.5 and 0.625 bits/sample on the Synthetic Source. Numbers in parentheses denote actual bit rate.

	VFS-UTSVQ		
	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR
8	8.21 (0.375)	9.71 (0.50)	10.62 (0.620)
16	8.36 (0.375)	9.86 (0.50)	10.78 (0.625)
32	8.38 (0.375)	10.00 (0.50)	10.95 (0.625)
UTSVQ	6.96 (0.375)	8.62 (0.55)	9.06 (0.605)

Table 2.5: Performance of VFS-UTSVQ at $b = 0.375$, 0.5 and 0.625 bits/sample on the Synthetic Source. Numbers in parentheses denote actual bit rate.

	FS-LBGVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR	SQNR
8	3.64	5.55	7.29	8.90
16	3.75	5.86	7.68	9.57
32	3.89	6.16	8.70	10.06
LBG-VQ	2.45	4.32	5.84	7.31

Table 2.6: Performance of FS-LBGVQ and LBG-VQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence.

	FS-TSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR	SQNR
8	3.19	5.15	6.81	8.29
16	3.46	5.42	7.12	8.73
32	3.59	5.53	7.33	9.06
TSVQ	2.18	3.59	5.24	6.62

Table 2.7: Performance of FS-TSVQ and TSVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence.

	VFS-LBGVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR	SQNR
8	6.09 (0.24)	6.96 (0.31)	10.45 (0.47)	11.73 (0.58)
16	6.46 (0.25)	8.64 (0.38)	10.85 (0.52)	12.02 (0.64)
32	7.34 (0.25)	8.76 (0.34)	11.52 (0.50)	12.04 (0.58)

Table 2.8: Performance of VFS-LBGVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence. Numbers in parentheses denote actual bit rate.

	VFS-TSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR	SQNR
8	5.86 (0.25)	6.56 (0.32)	9.49 (0.46)	10.54 (0.58)
16	6.23 (0.25)	7.98 (0.37)	10.83 (0.49)	12.09 (0.61)
32	6.33 (0.25)	9.12 (0.37)	11.63 (0.53)	12.69 (0.64)

Table 2.9: Performance of VFS-TSVQ at $b =$ 0.25, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence. Numbers in parentheses denote actual bit rate.

	VFS-UTSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR	SQNR
8	6.95 (0.25)	9.34 (0.375)	11.93 (0.49)	13.12 (0.625)
16	7.27 (0.25)	10.06 (0.385)	12.82 (0.49)	13.59 (0.625)
32	8.41 (0.25)	10.50 (0.342)	12.94 (0.49)	13.71 (0.591)
UTSVQ	4.58 (0.29)	5.50 (0.335)	7.32 (0.47)	9.95 (0.625)

Table 2.10: Performance of VFS-UTSVQ at $b =$ 0.25, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence. Numbers in parentheses denote actual bit rate.

	FS-LBGVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR	SQNR
8	2.13	3.50	4.90	5.98
16	2.06	3.62	5.03	5.97
32	2.19	3.76	5.67	6.22
LBG-VQ	1.90	3.23	4.37	5.55

Table 2.11: Performance of FS-LBGVQ and LBG-VQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence.

	FS-TSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR	SQNR
8	1.86	3.20	4.42	5.31
16	1.74	3.23	4.42	5.57
32	1.83	3.22	4.30	5.52
TSVQ	1.50	2.50	3.59	4.74

Table 2.12: Performance of FS-TSVQ and TSVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence.

	VFS-LBGVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR	SQNR
8	2.08 (0.17)	2.50 (0.21)	7.27 (0.39)	7.88 (0.51)
16	2.39 (0.18)	6.06 (0.33)	7.65 (0.47)	8.38 (0.52)
32	3.21 (0.19)	3.95 (0.25)	7.66 (0.42)	8.14 (0.51)

Table 2.13: Performance of VFS-LBGVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence. Numbers in parentheses denote actual bit rate.

	VFS-TSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR	SQNR
8	2.12 (0.18)	2.33 (0.22)	6.60 (0.39)	7.35 (0.51)
16	2.29 (0.18)	3.24 (0.26)	6.92 (0.40)	8.04 (0.53)
32	2.43 (0.19)	4.16 (0.26)	7.56 (0.43)	8.09 (0.55)

Table 2.14: Performance of VFS-TSVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence. Numbers in parentheses denote actual bit rate.

	VFS-UTSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
K	SQNR	SQNR	SQNR	SQNR
8	2.79 (0.18)	5.43 (0.32)	7.44 (0.40)	7.95 (0.52)
16	3.02 (0.18)	6.82 (0.35)	7.89 (0.41)	8.44 (0.54)
32	3.97 (0.19)	6.34 (0.26)	7.96 (0.41)	8.49 (0.49)
UTSVQ	2.29 (0.27)	3.03 (0.29)	4.41 (0.42)	5.82 (0.56)

Table 2.15: Performance of VFS-UTSVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence. Numbers in parentheses denote actual bit rate.

Chapter 3

2-D FSVQ and Image Coding Applications

Several schemes based on FSVQ have also been reported in the image coding literature. In [20] and [21], FSVQ was used to encode still images where the state was used to exploit the correlation in the spatial domain; over 50% saving in bit rate is achieved over LBG-VQ. In [22] and [23], FSVQ was used in coding image sequences where the state is defined to exploit temporal correlation; again a saving of over 50% is achieved over intraframe LBG-VQ.

In this chapter, we consider 2-dimensional (2-D) extensions of various FSVQ systems described in Chapter 2 and their application to low bit rate image coding. In particular, we describe two low bit rate image coding systems based of FSVQ. In one of the systems, we first subtract the mean of each input block and then encode the mean-subtracted block using 2-D extensions of fixed-rate and variable-rate FSVQ described in [18], [46]; the block mean is encoded separately. In the second system, a predictor is used to make a prediction of each block based on the previously encoded blocks [47]; the prediction residual (error) is again encoded using the 2-D versions of the fixed-rate and variable-rate FSVQ. It is shown by means of simulations that the image coding systems using variable-rate FSVQ give good performance results at low bit rates of 0.25-0.30 bits/pixel.

The rest of the chapter is organized as follows. In Section 3.1, we consider

2-D extensions of fixed-rate and variable-rate FSVQ and describe two low bit rate image coding systems based on 2-D FSVQ systems in Section 3.2. Section 3.3 provides the simulation results on images and finally a summary and conclusions are given in Section 3.4.

3.1 2-D FSVQ

In this section, we consider the 2-D extensions of the FSVQ schemes. We first describe the 2-D versions of fixed-rate FSVQ and their design algorithm and then proceed to discuss the 2-D extensions of variable-rate FSVQ.

3.1.1 Extension of Fixed-Rate FSVQ to 2-D

In the case of a 2-D source such as an image, each input vector is typically a 2-D block (matrix of size $l \times l$) and unlike the 1-D case, it has more than one adjacent preceding neighbor. For efficient encoding of an input vector $\mathbf{x}_n$, it is essential to exploit the correlation with the adjacent vectors in the north ($\mathbf{x}_{n-L}$) and west ($\mathbf{x}_{n-1}$) directions, where L is the number of blocks in an image row (see Fig. 3.1). In an FSVQ system, this is done by appropriately defining the state variable. To define the state variable, we associate with each input vector $\mathbf{x}_n$ an index $v_n \in \{1, 2, \ldots, K\}$ [1] and say that $\mathbf{x}_n$ is in "substate" v_n iff v_n is the index associated with $\mathbf{x}_n$. We then define the state $\mathbf{s}_n$ associated with $\mathbf{x}_n$ as a two-component vector [2] $\mathbf{s}_n = (v_{n-1}, v_{n-L})$, where v_{n-1} and v_{n-L} are, respectively, the substates associated with the west ($\mathbf{x}_{n-1}$) and north ($\mathbf{x}_{n-L}$) neighbors of $\mathbf{x}_n$. Note that $\mathbf{s}_n$ can be equivalently described by an index $k = K(v_{n-1} - 1) + v_{n-L}$ with $k \in \{1, 2, \ldots, K^2\}$.

[1] For instance, v_n can be the index of the the codevector in a size K codebook $\{\mathbf{c}(w), w \in \{1, 2, \ldots, K\}\}$ such that $\mathbf{x}_n$ is closest to $\mathbf{c}(v_n)$ in Euclidean distance. The precise definition of the substate is given in the next subsection.

[2] Since the vectors in the first row and first column do not have either a west or a north neighbor, we do not associate any state with these vectors and only define a substate for all such vectors.

With the state variable defined as above, an $(l \times l)$-dimensional $(K \times K)$-state FS-LBGVQ (FS-TSVQ) is specified by a state space $\mathcal{S} = \{1, 2, \ldots, K\} \times \{1, 2, \ldots, K\}$ and three mappings:

(1) $\alpha : \mathbb{R}^{l \times l} \times \mathcal{S} \rightarrow \mathcal{N}$: finite-state encoder,

(2) $\beta : \mathcal{N} \times \mathcal{S} \rightarrow \hat{\mathcal{A}}$: finite-state decoder,

(3) $\mathbf{f} \triangleq (f_1, f_2)$; f_i: $\mathcal{N} \times \mathcal{S} \rightarrow \{1, 2, \ldots, K\}$, $i = 1, 2$: next state function.

Here, $\mathcal{N} \triangleq \{1, 2, \ldots, N\}$ is the channel alphabet of size N and $\hat{\mathcal{A}}$ is the reproduction space. This definition is a 2-D extension of the 1-D FS-LBGVQ (FS-TSVQ) given in [18], [46]. Note that the next-state map $\mathbf{f}$ has two components f_1 and f_2 determining the substates associated with the west and north neighbors, respectively. The collection $\mathcal{C}_k \triangleq \{\beta(u, k), u \in \mathcal{N}\}$ is the codebook associated with state k; obviously, $\hat{\mathcal{A}} = \bigcup_{k=1}^{K^2} \mathcal{C}_k$.

Let $\{\mathbf{x}_n\}_{n=0}^{\infty}$ denote the input vector sequence, where $\mathbf{x}_n \in \mathbb{R}^{l \times l}$ is obtained from sampled image data. Similarly, let $\{u_n\}_{n=0}^{\infty}$, $\{\mathbf{s}_n\}_{n=0}^{\infty}$ and $\{\hat{\mathbf{x}}_n\}_{n=0}^{\infty}$ denote the channel symbol sequence, state sequence and reproduction vector sequence, respectively. With the substate sequence of the first row vector and the first column vector of the image given, the input process determines the sequence of channel symbols, reproduction vectors and states according to:

$$u_n = \alpha(\mathbf{x}_n, \mathbf{s}_n), \tag{3.1}$$

$$\hat{\mathbf{x}}_n = \beta(u_n, \mathbf{s}_n), \tag{3.2}$$

$$\mathbf{s}_{n+1} = (f_1(u_n, \mathbf{s}_n), f_2(u_{n+1-L}, \mathbf{s}_{n+1-L})),$$
$$n = m(L+1), m(L+2), \ldots, \; m = 1, 2, \ldots. \tag{3.3}$$

The rate of an FS-LBGVQ (FS-TSVQ) is given by $R = \log_2 N$, bits/vector. The FS-LBGVQ encoder is specified by the minimum distortion rule

$$\alpha(\mathbf{x}, k) = \arg\min_{u \in \mathcal{N}} d(\mathbf{x}, \beta(u, k)), \forall k \in \{1, 2, \ldots, K^2\}, \tag{3.4}$$

where d is the distortion measure, while for FS-TSVQ the encoding is done in a tree-structured manner.

3.1.2 Design Algorithm for 2-D Fixed-Rate FSVQ

Given a training sequence $\{\mathbf{x}_n,\ n = 0, 1, \ldots\}$ and a distortion measure, the design algorithm for an $(l \times l)$-dimensional, $(K \times K)$-state FS-LBGVQ (FS-TSVQ) of rate $R = \log_2 N$, bits/vector consists of designing:

(a) the state codebooks $\mathcal{C}_k$ for FS-LBGVQ (FS-TSVQ), each of size N, and

(b) the next-state function $\mathbf{f}(u, k), u \in \mathcal{N},\ k \in \{1, 2, \ldots, K^2\}$.

The design algorithm is an extension of the 1-D algorithm given in Chapter 2 and can be described in the following four steps.

1. Design an LBG-VQ with K codevectors for the given training sequence. We refer to this VQ as the substate-label VQ, $\mathcal{C} = \{\mathbf{c}(v), v \in \{1, 2, \ldots, K\}\}$. We say that an input vector $\mathbf{x}_n$ is associated with an ideal substate k_1 if

$$k_1 = \arg \min_{v \in \{1,2,\ldots,K\}} d(\mathbf{x}_n, \mathbf{c}(v)). \tag{3.5}$$

2. For each state index $k \in \{1, 2, \ldots, K^2\}$ of the FS-LBGVQ (FS-TSVQ), design an initial reproduction codebook $\mathcal{C}_k = \{\beta(u, k), u \in \mathcal{N}\}$ using the LBG algorithm [4] on the subtraining sequence $\{\mathbf{x}_n : k_1 = \arg\min_{v \in \{1,2,\ldots,K\}} d(\mathbf{x}_{n-1}, \mathbf{c}(v))$ and $k_2 = \arg\min_{v \in \{1,2,\ldots,K\}} d(\mathbf{x}_{n-L}, \mathbf{c}(v))\}$, where (k_1, k_2) is the pair associated with k (i.e., $k = K(k_1 - 1) + k_2$). Thus each codebook $\mathcal{C}_k$ is designed to be good for vectors whose west and north neighbors are associated with substates k_1 and k_2, respectively.

3. In order to enable the decoder to track the state sequence, we let the state sequence depend on the reproduction vector of the input vector rather than on the input vector itself. Thus given the substate-labels $\mathbf{c}(v)$ and the decoder

β designed in step (2), we define the mappings f_1 and f_2 and hence the next-state function $\mathbf{f} = (f_1, f_2)$ by [3]

$$f_1(u,k) = f_2(u,k) = \arg \min_{v \in \{1,2...,K\}} d(\beta(u,k), \mathbf{c}(v)),\ k \in \{1,2,\ldots,K^2\}, u \in \mathcal{N}. \tag{3.6}$$

4. Attempt to improve the state codebooks $\{\mathcal{C}_k, k \in \{1,2,\ldots,K^2\}\}$ of the FS-LBGVQ (FS-TSVQ) by encoding the training sequence using the next-state function obtained in step (3) and updating each codevector by replacing it by the centroid of the cell associated with that codevector. Also update the substate-label VQ $\mathcal{C}$ similarly.

3.1.3 Description of 2-D Variable-Rate FSVQ

The 2-D FS-LBGVQ (FS-TSVQ) system discussed so far is a fixed-rate system. The 2-D version of variable-rate FSVQ is specified by a state space $\mathcal{S} = \{1,2,\ldots,K\} \times \{1,2,\ldots,K\}$ and three mappings as follows:

(1) $\alpha : \mathbb{R}^{l \times l} \times \mathcal{S} \rightarrow \mathcal{N}(\mathcal{S})$: finite-state encoder,

(2) $\beta : \mathcal{N}(\mathcal{S}) \times \mathcal{S} \rightarrow \hat{\mathcal{A}}$: finite-state decoder,

(3) $\mathbf{f} \triangleq (f_1, f_2)$; f_i: $\mathcal{N}(\mathcal{S}) \times \mathcal{S} \rightarrow \{1,2,\ldots,K\}$, $i = 1,2$: next state function.

Here the channel alphabet depends on the state ($\mathcal{N}(k)$ for state k) and, in general, is different for each state. The rate of the system is given by $R = \sum_{k=1}^{K^2} P_k b_k$, bits/vector, where b_k is the average bit rate of the vector quantizer associated with state k and P_k is the probability of occurrence of state k. As in the 1-D case, the state vector quantizers can be LBG-VQ, TSVQ or UTSVQ.

Let D_{k,b_k} denote the average distortion incurred in state k when the rate of the quantizer associated with state k is b_k bits/vector. Then we wish to minimize the

[3] for FS-TSVQ, the encoding is done in a tree-structured manner.

average distortion given by

$$D = \sum_{k=1}^{K^2} P_k D_{k,b_k}, \tag{3.7}$$

subject to a constraint on the average rate described by

$$\sum_{k=1}^{K^2} P_k b_k \leq b_{avg}, \tag{3.8}$$

by appropriately designing the the bit assignment map $(b_1^*, b_2^*, \ldots, b_{K^2}^*)$, the state codebooks $\{\mathcal{C}_k, k \in \{1, 2, \ldots, K^2\}\}$ and the next-state function. The design algorithms given for variable-rate FSVQ for the 1-D case can be easily extended to 2-D in a manner similar to the case of fixed-rate FSVQ. The details are omitted here.

While the 1-D versions of FSVQ have been successfully applied to encoding of sampled speech [18], [46], direct application of 2-D versions of FSVQ on image data presents certain design problems. In what follows, we describe these problems and propose two methods to tackle them.

3.2 System Description

When the LBG algorithm [4] is used to design a small-sized LBG-VQ using a training sequence of images, the majority of the codevectors in the LBG-VQ correspond to the constant background vectors of different grey-levels and other feature vectors such as the edge vectors are either averaged out or masked by the background vectors [48]. Therefore in step (1) of the design algorithm for 2-D fixed-rate FSVQ, the majority of the substate-labels (i.e., the codevectors in the substate-label VQ) will correspond to the constant background vectors and the states will then correspond to the various combinations of constant background.

To account for the various grey-levels of background requires a very large number of states. To alleviate this problem and more importantly to avoid the masking of edge vectors by background vectors, we have considered two methods. In the first method we subtract off the block-mean from each input vector and encode

the block-mean and the residual separately [48]. Since the block-means of adjacent vectors are highly correlated, they can be encoded with a small number of bits; the residual is encoded using 2-D FS-LBGVQ and VFS-UTSVQ. We will use ME-FS-LBGVQ and ME-VFS-UTSVQ to denote, respectively, the systems in which FS-LBGVQ and VFS-UTSVQ are used to encode the residual signal. In the second method we use a predictor [47] to make a prediction of each pixel of the input vector based on the knowledge of the already encoded neighboring vectors and then encode the residual using the above mentioned FSVQs. We will refer to the system using FS-LBGVQ by PR-FS-LBGVQ, while PR-VFS-UTSVQ will be used for the system with VFS-UTSVQ.

As a result of block-mean subtraction or prediction, all constant background vectors will result in residual vectors close to the zero grey-level and they can be classified by using fewer number of states. The rest of the states can be devoted to other feature vectors. As compared to other vectors, the residual vectors corresponding to the near zero grey-level background can be encoded using relatively smaller number of bits while achieving comparable distortion making the residual vector sequence more amenable to variable-length coding. Details of the two systems are provided in the following subsections.

3.2.1 Encoding of Block-Mean

Typical real-world images exhibit high pixel-to-pixel correlation. For a vector of size 4×4 (the vector size considered throughout this thesis for images), the block-mean of an input vector is also highly correlated with the block-mean of the adjacent vectors. As a consequence, the block-mean can, in turn, be efficiently encoded by a VQ. However, due to the complexity associated with the VQs, we cannot design a very large size codebook. For instance, in order to achieve a rate of 1 bit/sample (for blocks of size 4×4 of block-means), the number of

codewords needed is 2^{16}, which is prohibitively large. To alleviate this problem, the block-mean is encoded in two steps: First, 4×4 blocks of block-mean are vector quantized using an LBG-VQ of small size and then the difference is encoded using an entropy-constrained block transform coding system. The entropy-constrained block transform coding system is similar to the system described in [49]. The only difference lies in the computation of the variances associated with the transform coefficients. In the present system depending on the codevector used to encode the block-mean vector we have a set of variances of the transform coefficients estimated by using a training sequence different from the test sequence.

3.2.2 Description of the Predictor

We used the 5^{th}-order predictor proposed in [47] to make the block prediction. Each pixel Y in an input vector (Fig. 3.2) is predicted using only five other pixels. The predicted value of Y is given by $\hat{Y} = \mathbf{aX}$, where $\mathbf{X} = [X_1, X_2, X_3, X_4, X_5]$. X_1 and X_2 are the two closest pixels in the same row of the vector in the west, X_3 is the lower right pixel of the upper left diagonal subblock and X_4 and X_5 are the two closest pixels in the same column of the north neighbor vector. The vector of linear prediction coefficients $\mathbf{a}$ is chosen to minimize the mean squared-error between Y and $\hat{Y}$ according to $\mathbf{a} = \mathbf{R}^{-1}\mathbf{d}$ [50]. Here $\mathbf{R} = E(\mathbf{X}^{\mathrm{T}}\mathbf{X})$ and $\mathbf{d} = E(\mathbf{X}Y)$. Spatial stationarity is assumed for computing the correlation matrix. We use the actual values of $\mathbf{X}$ to compute $\mathbf{a}$ but once $\mathbf{a}$ is computed $\hat{Y}$ is always estimated based on the reconstructed version of $\mathbf{X}$ as in any DPCM system.

3.2.3 Encoding of the Residual Signal

We have used FS-LBGVQ and VFS-UTSVQ for encoding the residual signal $\{\tilde{\mathbf{x}}_n\}$ obtained after block-mean subtraction or using 5^{th}-order prediction. When an input vector is encoded by means of an FSVQ, both linear and nonlinear depen-

dence of the input vector on its neighbors can be exploited by the FSVQ. The use of block-mean subtraction or a linear predictor removes linear dependence of the input vector on its neighbors to a certain extent; however the residual vector still retains the nonlinear dependence and even some linear dependence (due to the use of a coarse predictor to keep overhead information low) on its neighbors that can be efficiently exploited by the use of FSVQ on the residual sequence.

The FS-LBGVQ is designed for $\{\tilde{\mathbf{x}}_n\}$ by using the algorithm described in Subsection 3.1.2 and the encoding is done according to Equations (3.1), (3.2) and (3.3). The simulation results are presented in the next section.

As for the VFS-UTSVQ, we have considered two kinds of UTSVQs. In the first case, we design a complete TSVQ and then use the optimal pruning algorithm of [24] to obtain the UTSVQ of the desired rate. In the second case, we use the greedy algorithm of [51] to grow an unbalanced tree (instead of a complete TSVQ of the same rate) and then use the pruning algorithm of [24] to design the UTSVQ. For the reasons described in [51], the second type of UTSVQ, in general, gives better performance. We will refer to the ME-VFS-UTSVQ system based on the first and the second kind of UTSVQs by ME-VFS-UTSVQ1 and ME-VFS-UTSVQ2, respectively. Corresponding names for the prediction based systems are PR-VFS-UTSVQ1 and PR-VFS-UTSVQ2, respectively. The simulation results for various systems are given in the next section.

3.3 Simulation Results on Images

The performance results are reported in terms of the peak signal-to-noise ratio (PSNR) in dB and the overall average bit rate per pixel (including the overhead) denoted by b. The design values selected were $l = 4$ and $K = 4$. This corresponds to a vector dimension of 16 and the number of states is 16. The database used for training various kinds of FSVQ systems consisted of 19 monochrome images of

size 480×512 and the test image chosen was the 512×512 version of "Lena" not included in the training sequence.

3.3.1 Performance Results of ME-FS-LBGVQ and ME-VFS-UTSVQ

The VQs designed using the residual training sequence obtained by subtracting off the block-mean from each input vector will have zero-mean codevectors. As a consequence, for the squared-error distortion measure, it is easy to check that the overall distortion of ME-FS-LBGVQ and ME-VFS-UTSVQ systems is equal to the sum of distortions that result from encoding the block-mean and the residual signal. Therefore, the problem of encoding the block-mean and that of encoding the residual can be treated separately.

The block-mean was encoded using the system that consisted of an LBG-VQ followed by entropy-constrained block transform coding described briefly in the Subsection 3.2.1. The block-mean corresponding to the adjacent vectors are grouped together in block sizes of 4×4 and vector quantized with an LBG-VQ consisting of 8 codevectors. The remaining bits were allocated to entropy-constrained block transform coding. We allocated $\frac{1.0}{16}$ to $\frac{2.0}{16}$ bits/pixel to encode the block-mean. For smaller values of b, the bit rate allocated for block-mean encoding was kept close to $\frac{1.0}{16}$ bits/pixel since any further increase in the bit rate for block-mean encoder resulted in a relatively small decrease in the overall distortion as the major contribution to the overall distortion came from the encoding of the residual. On the other hand, for higher values of b, distortion arising out of the encoding of the residual signal is reduced substantially and the distortion resulting from block-mean encoding starts playing an equally important role in contributing to the overall distortion. As a consequence, the bit rate associated with the block-mean encoder is increased to as much as $\frac{2.0}{16}$ for $b = 0.38$ bits/pixel; any further

increase in the bit rate for block-mean encoding results in a very small decrease in overall distortion.

The residual sequence was encoded using FS-LBGVQ and VFS-UTSVQ. Table 3.1 illustrates the results for various systems at different bit rates. The term in the parentheses is the value of the bit rate per pixel actually achieved by the system, while the desired value is given by b. We have also provided the results for the case when the residual sequence is encoded using an LBG-VQ (ME-VQ) for comparison purposes. Table 3.1 shows that on the average, ME-FS-LBGVQ performs better than ME-VQ by about 0.8 dB at all bit rates shown in the table and ME-VFS-UTSVQ1 improves the performance by about 1 dB over ME-FS-LBGVQ. The best performance is attained by ME-VFS-UTSVQ2; it outperforms ME-VQ by about 3 dB at $b = 0.31$ bits/pixel achieving a PSNR of 31.66 dB. The reconstructed image at $b = 0.31$ bits/pixel using ME-VFS-UTSVQ2 is shown in Fig. 3.3.b; Fig. 3.3.a shows the original image.

3.3.2 Performance Results of PR-FS-LBGVQ and PR-VFS-UTSVQ

The PR-VFS-UTSVQ1 system is basically a more general version of the scheme described in [47]. In both these systems, an error sequence is formed by using a 5^{th}-order predictor. In [47], the error (residual) sequence is encoded using a UTSVQ obtained by optimally pruning a complete TSVQ [24]; this system can be looked upon as PR-VFS-UTSVQ1 with just one state.

We have considered the encoding of the residual using LBG-VQ, FS-LBGVQ and the two types of VFS-UTSVQ. The results are summarized in Table 3.2. We have also simulated and included the results of the system described in [47] (denoted by RDG) for comparison. The bit rate b here also includes the overhead information which consists of transmitting the actual value of each input sample in

the first row vector and the first column vector of the image (about $\frac{0.125}{16}$ bits/pixel). Table 3.2 shows that use of PR-FS-LBGVQ as opposed to PR-VQ leads to an increase of 0.6 dB in PSNR. For $b = 0.20$ and 0.26 bits/pixel, PR-VFS-UTSVQ1 outperforms PR-FS-LBGVQ by over 1.5 dB. However, this gain is reduced at higher rates. This reduction in gain can be attributed to the limitation posed by the size of the training sequence used. As opposed to the ME-VFS-UTSVQ systems, PR-VFS-UTSVQ2 shows an insignificant improvement over PR-VFS-UTSVQ1 system. Both PR-VFS-UTSVQ1 and PR-VFS-UTSVQ2 systems perform better than the system described in [47] at the rates shown in the table and the improvement is most noticeable at $b = 0.20$ and 0.26 bits/pixel, both visually and in terms of PSNR. Fig. 3.3.c shows the reconstructed image for PR-VFS-UTSVQ2 at $b = 0.32$ bits/pixel. The performance of PR-VFS-UTSVQ systems saturates above $b = 0.32$ bits/pixel due to the training sequence size constraint and therefore we have not included the results for higher rates in the table.

3.4 Summary and Conclusions

In this chapter, we have considered several variations of two low bit rate image coding systems based on fixed-rate and variable-rate FSVQs. Design algorithm for various schemes are obtained by appropriately modifying the algorithms given in chapter 2. None of these algorithms converge and there is no guaranteed improvement at each step of iteration. As a consequence, we performed some fixed number of iterations and chose the best case result. It can be concluded from our results that, in general, the variable-rate versions of FSVQ lead to performance improvements which in certain cases are quite substantial. The best performance was achieved by the systems based on variable-rate FSVQ using UTSVQs as the state quantizers for image coding just as in the case of speech coding.

For both systems, we restricted ourselves to 16 states corresponding to the

choice of $K = 4$. In order to estimate the gain obtained by increasing the number of states, we considered $K = 6$ and 7 and observed that the performance gain was relatively small (of the order of 0.2 to 0.3 dB in PSNR value) over the case of 16 states.

For image coding, out of all the schemes considered, PR-VFS-UTSVQs have the potential of doing the best. For example, at 0.25 bits/pixel, a PSNR of 30.74 dB is achieved for encoding the 512×512 version of "Lena" and the quality of reconstructed image is good. Although, the PSNR value achieved at $b = 0.31$ bits/pixel was 31.2 dB, they have the potential to do better; we believe that the gain in the PSNR value at $b = 0.31$ bits/pixel over that at $b = 0.25$ bits/pixel can be higher if we use a larger training sequence.

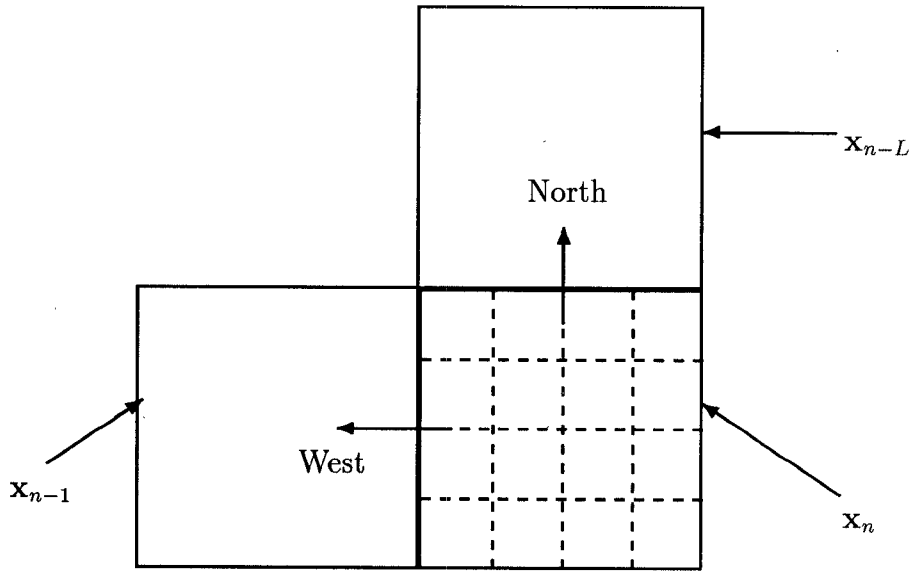


Figure 3.1: Input vector $\mathbf{x}_n$ and its north ($\mathbf{x}_{n-L}$) and west ($\mathbf{x}_{n-1}$) neighbors.

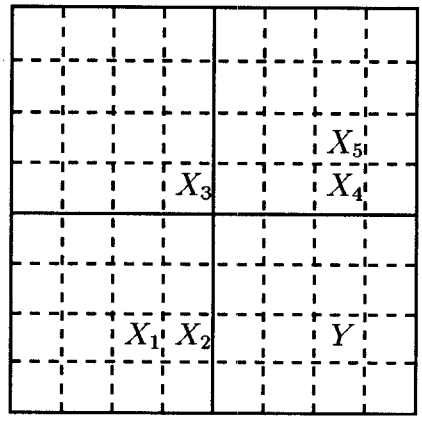


Figure 3.2: The predictor structure.

(a)

(b)

(c)

Figure 3.3: (a) Orignal, (b) ME-VFS-UTSVQ2 ($b = 0.31$), (c) PR-VFS-UTSVQ2 ($b = 0.32$)

	ME-VQ	ME-FS-LBGVQ	ME-VFS-UTSVQ1	ME-VFS-UTSVQ2
b	PSNR	PSNR	PSNR	PSNR
0.19	27.35	27.99	28.64	28.73
	(0.19)	(0.19)	(0.20)	(0.20)
0.25	28.02	28.83	29.75	30.31
	(0.25)	(0.25)	(0.26)	(0.27)
0.31	28.58	29.56	30.75	31.66
	(0.31)	(0.31)	(0.32)	(0.32)
0.38	29.13	30.16	31.67	32.00
	(0.38)	(0.38)	(0.38)	(0.39)

Table 3.1: Performance of ME-VQ, ME-FS-LBGVQ, ME-VFS-UTSVQ1 and ME-VFS-UTSVQ2 at $b = 0.19,\ 0.25,\ 0.31$ and 0.38 bits/pixel on the 512×512 version of Lena. Numbers in parentheses denote actual bit rate.

	PR-VQ	PR-FS-LBGVQ	PR-VFS-UTSVQ1	PR-VFS-UTSVQ2	RDG
b	PSNR	PSNR	PSNR	PSNR	PSNR
0.20	27.22	27.89	29.70	29.86	29.16
	(0.20)	(0.20)	(0.20)	(0.20)	(0.20)
0.26	28.62	29.11	30.64	30.74	29.80
	(0.26)	(0.26)	(0.25)	(0.25)	(0.25)
0.32	29.50	30.16	31.22	31.19	31.00
	(0.32)	(0.32)	(0.32)	(0.32)	(0.32)

Table 3.2: Performance of PR-VQ, PR-FS-LBGVQ, PR-VFS-UTSVQ1, PR-VFS-UTSVQ2 and RDG at $b = 0.20,\ 0.26$, and 0.32 bits/pixel on the 512×512 version of Lena. Numbers in parentheses denote actual bit rate.

Chapter 4

Finite-State Vector Quantization for Noisy Channels

The main reason for the superior performance of FSVQ as compared to the ordinary VQ is that FSVQ effectively performs a two stage quantization as opposed to the single stage operation of VQ; the first stage of FSVQ does a rough quantization based on the state index (at no extra overhead since the state index is determined based on the previous output and state) and the second stage of FSVQ performs a finer quantization while operating at the same rate as VQ. However, it is only under noiseless channel conditions that the decoder can track the encoder state sequence. In the case when the channel is noisy, even a single error in the transmitted encoder output can lead to an erroneous decoder state sequence for several time indices leading to a rapid degradation in the performance of FSVQ. In this chapter we study the performance of FSVQ in the presence of channel noise and devise new FSVQ designs for operation over a noisy channel.

In the past, combined source-channel coding has been studied in various quantization contexts. Design algorithms for an optimum scalar quantizer operating over a noisy channel were developed in [25], [26]. These ideas were extended to full-searched VQ (LBG-VQ) in [27] and then to tree-structured VQ (TSVQ) and multi-stage VQ (MSVQ) in [28]. Furthermore, necessary conditions of optimality for trellis encoding systems have been developed in [29] and a design algorithm

based on these necessary conditions is described in [30]. An FSVQ can be looked upon as a trellis encoding system with unit search depth and a general next-state function (as opposed to the one imposed by the shift-register implementation). The problem of combined source-channel coding for FSVQ can be formulated along the lines of the system in [29]. However, due to the use of a general next-state function, the approach of [29] for the trellis encoding system cannot be used as such. We shall elaborate on this in Section 4.1.3. As a consequence, we make simplifying assumptions about the FSVQ structure and describe two noisy channel FSVQ systems (NC-FSVQ1 and NC-FSVQ2). In NC-FSVQ1, we assume that the "protected" encoder state is transmitted periodically to the decoder, while in NC-FSVQ2, we try to modify the FSVQ structure in such a manner that without losing too much in terms of performance (as compared to FSVQ of [18] in noiseless case), the theory of [29] can be used after some modifications; in particular we design the FSVQ such that the next state information is completely contained in the first few most significant bits of the codeword.

One important application of NC-FSVQ1 and NC-FSVQ2 that we will consider in this chapter is that of quantizing speech LSP parameters [34]. Currently, there is a growing interest in encoding speech LSP parameters [31]-[33] used in vocoders [34] and hybrid speech coders [35]-[38]. In several applications, it is necessary that the speech coder be robust against transmission errors. An important application is digital cellular networks where large channel error rates may arise from various sources such as multipath fading and interference from other channels [52]. We will show that the noisy channel FSVQ systems proposed here present an interesting alternative for such situations. They efficiently utilize the interframe and intraframe correlation of LSP parameters while providing a fairly robust performance in the presence of channel noise. We will see that if the channel is noisy and a delay of 4-8 frames can be tolerated then NC-FSVQ1 is the right candidate; on

the other hand NC-FSVQ2 will be the right choice for a low delay system (delay of 1 frame).

The rest of the chapter is organized as follows. In Section 4.1, we briefly describe the FSVQ of [18], state the problems associated with it in the presence of channel noise and formulate the problem of designing FSVQ for noisy channels. In Sections 4.2 and 4.3, the description and design algorithms of NC-FSVQ1 and NC-FSVQ2, respectively, are provided. In Section 4.4 simulation results for the Gauss-Markov source and speech LSP parameters are presented followed by a summary and conclusions in Section 4.5.

4.1 Preliminaries

4.1.1 Definition of FSVQ

In this section, we recapitulate the definition of FSVQ. An L-dimensional K-state FSVQ [18] is specified by a state space $\mathcal{S} = \{1, 2, \ldots, K\}$, an initial state s_0 and the mappings:

(1) $\alpha : \mathbb{R}^L \times \mathcal{S} \rightarrow \mathcal{N}$: finite-state encoder,

(2) β: $\mathcal{N} \times \mathcal{S} \rightarrow \hat{\mathcal{A}}$: finite-state decoder,

(3) f: $\mathcal{N} \times \mathcal{S} \rightarrow \mathcal{S}$: next state function.

Here, $\mathcal{N} \triangleq \{1, 2, \ldots, N\}$ is the finite channel alphabet of size N and $\hat{\mathcal{A}}$ is the reproduction space.

Let $\{\mathbf{x}_n\}_{n=0}^{\infty}$ denote the input vector sequence, where $\mathbf{x}_n \in \mathbb{R}^L$. Similarly, let $\{u_n\}_{n=0}^{\infty}$, $\{s_n\}_{n=0}^{\infty}$ and $\{\hat{\mathbf{x}}_n\}_{n=0}^{\infty}$ denote the channel symbol sequence, state sequence and reproduction vector sequence, respectively. With initial state s_0, the input sequence determines the sequence of channel symbols, reproduction vectors and states according to:

$$u_n = \alpha(\mathbf{x}_n, s_n), \tag{4.1}$$

$$\hat{\mathbf{x}}_n = \beta(u_n, s_n), \tag{4.2}$$

$$s_{n+1} = f(u_n, s_n), \quad n = 0, 1, \ldots. \tag{4.3}$$

The next state depends only on the present state and the output channel symbol; therefore, given the initial state and correct channel symbol sequence, the decoder can track the state sequence. Here, $\mathcal{C}_s \triangleq \{\beta(u, s), u \in \mathcal{N}\}$ is the codebook associated with state s and $\hat{\mathcal{A}} = \bigcup_{s=1}^{K} \mathcal{C}_s$. The rate of the FSVQ is given by $R = \log_2 N$, bits/vector. The encoder mapping is specified in terms of a distortion function that is used to measure the performance of the FSVQ. The distortion measure $d : \mathbb{R}^L \times \hat{\mathcal{A}} \rightarrow [0, \infty)$ assigns a non-negative cost $d(\mathbf{x}, \mathbf{y})$ to reproducing the input vector $\mathbf{x}$ as $\mathbf{y}$. The encoder is specified by the minimum distortion rule [18]

$$\alpha(\mathbf{x}, s) = \arg\min_{u \in \mathcal{N}} d(\mathbf{x}, \beta(u, s)), \forall s \in \mathcal{S}. \tag{4.4}$$

The average distortion incurred in an FSVQ system is given by $\frac{1}{L}E[d(\mathbf{X}, \mathbf{Y})]$, where the expectation is taken w.r.t. the source distribution.

An FSVQ can be looked upon as a time varying VQ. Depending on the past quantization characteristics, an FSVQ makes a prediction about the region in $\mathbb{R}^L$ where the current input vector may belong to and then performs a fine quantization over that region (using a VQ designed for that region). In this manner, the FSVQ is able achieve the efficiency of a high rate VQ at a relatively lower actual rate. Due to its built-in memory as opposed to the LBG-VQ which is memoryless, FSVQ is capable of exploiting inter-vector correlation leading to improved performance over the LBG-VQ.

4.1.2 Performance of the FSVQ in a Noisy Channel

Although an FSVQ system offers performance improvement over memoryless VQ by exploiting inter-vector correlation, it becomes prone to channel error propagation due to the presence of a feedback structure in the FSVQ encoder. In the

absence of channel noise, the feedback eliminates the need to transmit the encoder state to the receiver side. However, the ability of the decoder to track the encoder state sequence *critically* depends on the error free transmission of the codewords over the channel. Even a single error occurring in the transmitted codeword can lead to an incorrect decoder state. Once the decoder state becomes different from the encoder state, the decoder state sequence can remain derailed for a long time. In practice, since there is only a finite, and usually small number of states, the encoder and the decoder states become the same after some time. Despite the fact that the decoder is able to eventually track the encoder state, the performance degradation is severe because once the decoder loses track of the encoder state, its state sequence essentially becomes random and the input vectors get mapped to random reproduction vectors. This statement will be confirmed by the results illustrated in the simulation results section. In order to control the performance of FSVQ under noisy channel conditions, it is essential to avoid or minimize the "derailing" of the decoder. This can be achieved by either (i) transmitting some "protected version" of the encoder state sequence periodically or (ii) modifying the FSVQ system so that it becomes self-tracking (i.e., decoder can start tracking soon after derailing). We elaborate on these issues in Sections 4.2 and 4.3. In what follows, we formulate the FSVQ design problem under noisy channel conditions.

4.1.3 Formulation of the Problem

Consider the FSVQ block diagram in Fig. 4.1. The encoder α can be described in terms of the partition $\{\mathcal{P}_{s,u},\ s \in \mathcal{S}, u \in \mathcal{N}\}$ such that $\alpha(\mathbf{x}, s) = u$, if $\mathbf{x} \in \mathcal{P}_{s,u},\ s \in \mathcal{S}, u \in \mathcal{N}$. The output u of the encoder α is transmitted over a discrete memoryless channel (DMC) γ described by a random mapping $\gamma : \mathcal{N} \rightarrow \mathcal{N}$ and the transition matrix

$$Q(v|u) = Pr(\gamma(u) = v),\ \forall u, v \in \mathcal{N}, \tag{4.5}$$

where v is the output of the DMC. The decoder β maps the received codeword v into a reproduction vector $\mathbf{y}$ depending on its state $\hat{s}$. The average distortion incurred in the system of Fig. 4.1 is given by

$$D(\alpha, \beta, f) = \frac{1}{L} E[d(\mathbf{X}, \mathbf{Y})], \tag{4.6}$$

where the expectation is taken w.r.t the source and channel distributions. The problem is to minimize D by appropriate designs of α, β and f.

Next, we attempt to develop necessary conditions for the optimality of the system. First, note that the average distortion can be expressed as

$$\begin{aligned} D(\alpha, \beta, f) &= \frac{1}{L} \sum_{s,u} \sum_{\hat{s},v} E[d(\mathbf{X}, \mathbf{Y}) | s, \hat{s}, u, v] Pr(s, \hat{s}, u, v) & (4.7) \\ &= \frac{1}{L} \sum_{s,u} \int_{\mathcal{P}_{s,u}} \{ \sum_{\hat{s},v} Q(v|u) Pr(\hat{s}|s) d(\mathbf{x}, \beta(v, \hat{s})) \} p(\mathbf{x}) d\mathbf{x}. & (4.8) \end{aligned}$$

We denote the term in the braces in (4.8) by a cost function

$$\begin{aligned} C_s(\mathbf{x}, u) &= \sum_{\hat{s},v} Q(v|u) Pr(\hat{s}|s) d(\mathbf{x}, \beta(v, \hat{s})) & (4.9) \\ &= E_{\hat{S},V}[d(\mathbf{x}, \beta(V, \hat{S}) | s, u], & (4.10) \end{aligned}$$

where $\hat{S}$ and V denote the decoder state and the channel output random variables, respectively. Then

$$D(\alpha, \beta, f) = \frac{1}{L} \sum_{s,u} \int_{\mathcal{P}_{s,u}} C_s(\mathbf{x}, u) p(\mathbf{x}) d\mathbf{x}. \tag{4.11}$$

We first assume that f and β are fixed and try to find the optimal α or equivalently the optimal partition. Clearly, from Equation (4.11), the optimal partition must satisfy [29]

$$\mathcal{P}^*_{s,u} = \{\mathbf{x} : C_s(\mathbf{x}, u) \leq C_{s'}(\mathbf{x}, u'),\ \forall s' \in \mathcal{S}, u' \in \mathcal{N}\}, \tag{4.12}$$

with an additional constraint that the state sequence should lie on the trellis corresponding to the FSVQ. Similarly, for fixed α and f, the optimum β must satisfy

$$\beta^*(v, \hat{s}) = \arg \min_{\mathbf{y} \in \mathbb{R}^L} \sum_{s,u} Q(v|u) Pr(\hat{s}|s) \int_{\mathcal{P}_{s,u}} d(\mathbf{x}, \mathbf{y}) p(\mathbf{x}) d\mathbf{x},\ \hat{s} \in \mathcal{S}, v \in \mathcal{N}. \tag{4.13}$$

However, for fixed α and β, the optimum f is not easy to compute and we use *ad hoc* schemes to update f. This step leads to an overall suboptimal system just as in the case of FSVQ under noiseless channel assumptions in [18].

The computation of $C_s(\mathbf{x}, u)$, $s \in \mathcal{S}, u \in \mathcal{N}$ involves the estimation of a probability term $Pr(\hat{s}|s)$. For a general next-state function, it is not clear how the decoder state $\hat{s}$ and the encoder state s are related to each other at any given time and this makes the estimation of $Pr(\hat{s}|s)$ difficult. Next, we present two special cases where we make certain assumptions about the system that help in estimating the transition probability between the states and hence the cost function C_s.

4.2 Description and Design of NC-FSVQ1

In this section we consider the first special case in which we assume that the encoder state sequence is known to the decoder (by transmitting the protected state indices). As mentioned earlier, the FSVQ can be looked upon as a trellis encoding system [30] with unit search depth and a general next-state function (as opposed to one imposed by shift-register implementation). While for trellis encoding systems utilizing a shift-register implementation, it is straightforward to compute $Pr(\hat{s}|s)$, for the FSVQ with a general next-state function, this probability cannot be easily computed. So, we make a simplifying assumption that the encoder state sequence is protected and transmitted separately to the decoder. Assuming that $\hat{s} = s$ at all times, we have

$$\begin{aligned} C_s(\mathbf{x}, u) &= E_V[d(\mathbf{x}, \beta(V, s))|s, u] \\ &= \sum_v Q(v|u) d(\mathbf{x}, \beta(v, s)). \end{aligned} \tag{4.14}$$

The cost function in (4.14) for each state $s \in \mathcal{S}$ corresponds to the modified distortion defined to design a channel-optimized VQ (CO-VQ) in [27].

In an attempt to find the necessary conditions for the optimality of the FSVQ

system under noisy channel conditions (with the assumption that the encoder state sequence is available at the decoder), we proceed as follows. For fixed β and f, the optimum α is defined by the partition given by (4.12) with C_s given by (4.14). This exactly corresponds to determining the optimum partition for CO-VQ in [27] within each state $s \in \mathcal{S}$. Similarly for fixed α and f, the optimum β is obtained using (4.13) with $Pr(\hat{s}|s) = 1$, if $\hat{s} = s$ and $Pr(\hat{s}|s) = 0$, otherwise. Again this corresponds to determining the optimum reproduction vectors in [27] (for a fixed partition) within each state $s \in \mathcal{S}$. Finally, for fixed α and β, f is determined in an *ad hoc* fashion as described in the following design algorithm. The design algorithm for the noisy channel FSVQ can now be described by the following steps.

Algorithm:

1. Design an LBG-VQ [4] with K codevectors for the given training sequence. This is referred to as the state-label VQ, $\mathcal{C} = \{\mathbf{c}(s), s \in \mathcal{S}\}$.

2. For each state s, use the algorithm of [27] to design an initial reproduction codebook (CO-VQ) $\mathcal{C}_s = \{\beta(u,s), u \in \mathcal{N}\}$ on the subsequence composed of all successors to vectors for which the state-label VQ chooses s, i.e., the subsequence $\{\mathbf{x}_n : s = \arg\min_{k \in \mathcal{S}} d(\mathbf{x}_{n-1}, \mathbf{c}(k))\}$.

3. As in [18], we define a next-state function f by[1]

$$f(u,s) = \arg\min_{k \in \mathcal{S}} d(\beta(u,s), \mathbf{c}(k)),\ s \in \mathcal{S}, u \in \mathcal{N}. \tag{4.15}$$

4. Attempt to improve the state codebooks $\{\mathcal{C}_s, s \in \mathcal{S}\}$ by encoding (as in [27]) the training sequence using the next-state function obtained in step (3) and

[1]It might appear that since we are explicitly transmitting the encoder state to the decoder (i.e., the system is omniscient in the sense of [18]), it is unnecessary to define the next-state function based on the reproduction vectors. But as we will see later, this definition of next-state function helps in reducing overhead information.

updating each codevector by the generalized centroid of its corresponding cell [27]. Also update the state-label VQ $\mathcal{C}$.

The difference between the above algorithm and the one in [18] is that here the state codebooks are CO-VQs and the encoding is done as described in [27] within each state. Also, it is implicitly assumed that the decoder has perfect knowledge of the encoder state sequence. This implies that a "protected" encoder state sequence needs to be transmitted to the receiver. Since transmission of the protected encoder state for each vector can lead to a very large amount of overhead, the encoder state index is transmitted only periodically, say, every p frames; given the state indices at times k and $k+p$ (s_k and s_{k+p}) and the received codewords $\{v_i\}$ at times $i = k, \ldots, k+p-1$, a maximum likelihood (ML) estimate of the encoder state at times $i = k+1, \ldots, k+p-1$, is obtained at the decoder. Specifically, using the Viterbi algorithm [53], we find the most likely transmitted sequence $\{u_i^*\}_{i=k}^{k+p-1}$ satisfying

$$Pr(\{u_i^*\}_{i=k}^{k+p-1} | \{v_i\}_{i=k}^{k+p-1}, s_k, s_{k+p}) =$$
$$\max_{\{u_i\}_{i=k}^{k+p-1}} Pr(\{u_i\}_{i=k}^{k+p-1} | \{v_i\}_{i=k}^{k+p-1}, s_k, s_{k+p}). \tag{4.16}$$

The argument of the right hand side expression in (4.16) can be expressed as a product term given by

$$Pr(\{u_i\}_{i=k}^{k+p-1} | \{v_i\}_{i=k}^{k+p-1}, s_k, s_{k+p}) =$$
$$[\prod_{i=k}^{k+p-1} Pr(u_i | v_i, s_i)] Pr(u_{k+p-1} | v_{k+p-1}, \hat{s}_{k+p-1}, s_{k+p}) I_{s_{k+p}, f(v_{k+p-1}, \hat{s}_{k+p-1})}, \tag{4.17}$$

where $I_{s,s'}$ is an indicator function satisfying

$$I_{s,s'} = \begin{cases} 1 & \text{if } s = s' \\ 0 & \text{if } s \neq s' \end{cases} \quad \forall s, s' \in \mathcal{S}. \tag{4.18}$$

To derive the product form above, we used the fact that the channel is memoryless and that the decoder state at time $n+1$, $\hat{s}_{n+1}$, is uniquely determined by $\hat{s}_n$ and

v_n. Once the most likely transmitted sequence is identified, the estimated state sequence is obtained from $s_i^* = f(u_{i-1}^*, s_{i-1}^*)$, $i = k+1, \ldots, k+p-1$, with $s_k^* = s_k$. As can be seen from (4.17), the next-state function defined in step (3) of the design algorithm helps in estimating the transmitted codeword. Note that this procedure introduces a decoding delay of p frames.

In the above ML state estimation, there is a nonzero probability that some states are estimated incorrectly. However, the noisy channel FSVQ described so far assumes that the encoder and decoder are in the same state and only the codeword index can be corrupted by the channel noise. If the decoder state differs from the encoder state at a certain time instant, then the resulting error can be very large even if the codeword is received correctly. To reduce the resulting distortion in such situations, we perform a judicious indexing of the codevectors *among* the states. The basic idea is the following. The codevectors among the states that are close in the Euclidean sense are assigned binary codewords that are close in the Hamming sense; the highest priority is given to those states which are most likely to be confused with each other. The details of this algorithm can be found in the Appendix A. This algorithm leads to a robust performance especially when the period p is large in which case the probability of erroneous estimation of states is higher. In the sequel, we refer to this modified FSVQ system (with "protected" encoder state transmission, state estimation and codeword reassignment) as the noisy channel FSVQ (NC-FSVQ1). Note that under noiseless channel conditions, NC-FSVQ1 is identical to the noiseless channel FSVQ of [18] and it will be referred to as FSVQ1.

4.3 Description and Design of NC-FSVQ2

Although the NC-FSVQ1 described in the previous section offers robustness against channel noise especially for encoding speech LSP parameters as will be seen in Sec-

tion 4.4, there are two problems associated with such a scheme. First, it suffers from a delay at the receiver side. The decoder receives the protected encoder state say at times k and $k+p$ and then based on the received information, estimates the most likely encoder states at intermediate times $k+1, k+2, \ldots, k+p-1$. Thus the decoder has to wait till time $k+p$ to estimate the encoder state at time $k+1$, leading to a maximum delay of p vectors. Such delays may be unacceptable in certain low delay systems. The second problem relates to the overhead involved in explicitly transmitting the protected encoder state information; the next encoder state information is implicitly contained, at least partially, in the current transmitted codeword and it seems that protecting the codeword (or part of it) instead of the state (as is done in NC-FSVQ1) might lead to a similar performance (in terms of distortion) at a lower overhead rate.

Let us now focus on the second issue. In a given FSVQ, the state information is embedded in the codeword in an unstraightforward way. In other words, we do not know which bits in the codeword should be protected in an effort to effectively protect the state information. Also, as the rate of the system increases, the codeword length increases and in an effort to protect the state information (by protecting bits in the codeword), we may need to protect a larger number of bits at higher rates. Clearly, it would be desirable if the state information could be placed in some fixed positions of the transmitted codeword (e.g., in the first few most significant bits of the codeword assuming $R > \log_2 K$). Now suppose the first $l = \log_2 K$ bits of the codeword contain all the state information, i.e., if $u_n = (u_{n,1}, u_{n,2}, \ldots, u_{n,l}, \ldots, u_{n,N})$ is the binary codeword and s_n is the state at time n, the state at time $n+1$, s_{n+1}, is given by

$$\begin{aligned} s_{n+1} &= f(u_n, s_n) = f(u_{n,1}, u_{n,2}, \ldots, u_{n,l}, \ldots, u_{n,N}, s_n) \\ &= f(u_{n,1}, u_{n,2}, \ldots, u_{n,l}, s_n). \end{aligned} \tag{4.19}$$

Furthermore, if, in addition, we have

$$f(u,s) = f(u),\ \forall s \in \mathcal{S}, \tag{4.20}$$

then it would mean that in the presence of noise in the channel, the FSVQ will not derail indefinitely because as soon as the first l bits of the transmitted codeword are received correctly, the decoder will retrack the encoder state (since at that instant the decoder state becomes the same as the encoder state). Another advantage of such a system is that if the state information is to be protected, as in NC-FSVQ1, it will also help in receiving the codewords more reliably.

In what follows, we will show that it is possible to design an FSVQ system satisfying (4.19) and (4.20) by modifying the design algorithm of [18]. The modified design algorithm, leading to a scheme called FSVQ2, is provided next.

4.3.1 Design of FSVQ2 under Noiseless Conditions

Suppose for a given training sequence $\{\mathbf{x}_n\}$, we want to design a K-state FSVQ of rate R bits/vector ($2^R > K$) with a next-state function satisfying (4.19) and (4.20). Then the design algorithm proceeds as follows.

Algorithm:

1. Design an LBG-VQ [4] with K codevectors for the given training sequence. We refer to this VQ as the state-label VQ, $\mathcal{C} = \{\mathbf{c}(s), s \in \mathcal{S}\}$.

2. For each state s, design an initial reproduction codebook $\mathcal{C}_s = \{\beta_1(u^1, s), u^1 \in \mathcal{N}_1\}$ using the LBG algorithm [4] on the subtraining sequence composed of all successors to vectors for which the state-label VQ chooses s, i.e., the subsequence $\{\mathbf{x}_n : s = \arg\min_{k\in\mathcal{S}} d(\mathbf{x}_{n-1}, \mathbf{c}(k))\}$. Here we assume that $\mathcal{N}_1 = \mathcal{S}$ and therefore each state codebook consists of K (as opposed to 2^R codevectors in FSVQ of [18]) codevectors.

3. The ideal state s in step (2) depends on the input vector and therefore the decoder will not be able to track the ideal state sequence. In order to enable the decoder to track the state sequence, we choose the next state as the label which best matches the *reproduction* of the current input vector rather than the current input vector itself. Thus, given the state labels $\mathbf{c}(s)$ and the decoder β_1 designed in step (2), we define a next-state function f by

$$f(u^1, s) = \arg\min_{k \in \mathcal{S}} d(\beta_1(u^1, s), \mathbf{c}(k)),\ s \in \mathcal{S}, u^1 \in \mathcal{N}_1. \tag{4.21}$$

4. For each state $s \in \mathcal{S}$, we perform the following reindexing of all the codevectors in $\mathcal{C}_s$. First, we group all the codevectors in $\mathcal{C}_s$ that have the same value of $f(.,s)$ (i.e., they lead to the same next state, say, $\tilde{s}$). Then in each such group, the codevector with highest probability of occurrence is reassigned the common index $\tilde{s}$. The remaining codevectors from each group are finally reassigned, in an arbitrary fashion, a "unique" index in $\mathcal{N}_1$ which has not yet been assigned to any codevector. Note that after reassignment[2], if state s can transit to state s' then the next-state function will satisfy $f(s', s) = s'$.

5. In each state codebook $\mathcal{C}_s$, we retain only those codevectors whose indices satisfy $I_{u^1, f(u^1,s)} = 1$, $\forall u^1 \in \mathcal{N}_1$, where $I_{u^1, f(u^1,s)}$ is defined in (4.18). We denote the reduced version of $\mathcal{C}_s$ by $\mathcal{C}'_s$ which consists of $K_s = \sum_{u^1 \in \mathcal{N}_1} I_{u^1, f(u^1,s)}$ codevectors and use f' to denote the modified next-state function. The codewords of $\mathcal{C}'_s$ are binary tuples of length $l = \log_2 K$ with some of the l-tuples possibly not allowed (corresponding to the codevectors that are not retained and thereafter discarded). Now consider a K-state FSVQ with state codebooks $\mathcal{C}'_s$ and the next-state function f'. The rate of the system is $\log_2 K$ bits/vector and if the codeword transmitted at time n is u^1_n, then the state at time $n+1$, s_{n+1}, is $f'(u^1_n, s_n) = u^1_n$, irrespective of the state s_n.

[2]It should also be noted that under noiseless channel conditions, the reindexing of the codevectors will not affect the performance of the FSVQ designed using steps (1)-(3).

6. Attempt to improve the state codebooks $\{\mathcal{C}'_s, s \in \mathcal{S}\}$ of the FSVQ by encoding the training sequence using the next-state function obtained in step (5) and updating each codevector by replacing it by the centroid of the cell associated with that codevector. Also update the state-label VQ $\mathcal{C}$ similarly. Repeat steps (3)-(6) for some fixed number of iterations and choose the best case. The designed FSVQ clearly has a next-state function satisfying (4.19) and (4.20).

7. The FSVQ system designed so far has a rate $\log_2 K$ bits/vector. To operate at the desired rate of R bits/vector, we encode the whole training sequence using the FSVQ system $\{\{\mathcal{C}'_s\}_{s\in\mathcal{S}}, f'\}$ and then design[3] an LBG-VQ of rate $(R - \log_2 K)$ bits/vector for the subtraining sequence associated with each pair (s, u^1), where $s \in \mathcal{S}$ and u^1 is the codeword associated with codevectors in $\mathcal{C}'_s$.

The overall modified FSVQ system obtained using the above algorithm can be described by a block diagram given in Fig. 4.2. In the block diagram, the K-state FSVQ encoder is completely specified by the next-state function f' and a mapping α_1 which is described in terms of the partition $\{\mathcal{P}_{s,u^1}, s \in \mathcal{S}, u^1 \in \mathcal{N}_1\}$ according to

$$\alpha_1(\mathbf{x}, s) = u^1, \text{ if } \mathbf{x} \in \mathcal{P}_{s,u^1}. \tag{4.22}$$

We will refer to α_1 as the primary encoder. Recall that for the primary encoder the channel alphabet (output of the encoder) is the same as the state space $\mathcal{S}$.

The LBG-VQ encoder in Fig. 4.2 is specified completely by a mapping α_2 which

[3]Note that we are designing one LBG-VQ for each pair (s, u^1) in FSVQ2 (and its noisy version to be discussed in a later section) of rate $(R - \log_2 K)$ bits/vector as opposed to just one LBG-VQ of rate R bits/vector for each state in FSVQ1 (and its noisy version NC-FSVQ1); this is done to ensure that the memory requirements for FSVQ1 and FSVQ2 (and their noisy versions) are the same.

is described in terms of the partition $\{\mathcal{P}_{s,u^1}^{u^2}, s \in \mathcal{S}, u^1 \in \mathcal{N}_1, u^2 \in \mathcal{N}_2\}$ according to

$$\alpha_2(\mathbf{x}, s, u^1) = u^2, \text{ if } \mathbf{x} \in \mathcal{P}_{s,u^1}^{u^2}. \tag{4.23}$$

Here the channel space $\mathcal{N}_2$ is related to $\mathcal{N}$ and $\mathcal{N}_1$ by $\mathcal{N} = \mathcal{N}_1 \times \mathcal{N}_2$ and therefore the number of symbols N_2 in $\mathcal{N}_2$ is given by $N_2 = 2^R/K$. We also have

$$\bigcup_{u^2} \mathcal{P}_{s,u^1}^{u^2} = \mathcal{P}_{s,u^1}, \ \forall s \in \mathcal{S}, u^1 \in \mathcal{N}_1. \tag{4.24}$$

The FSVQ decoder and the LBG-VQ decoder can be jointly specified by f' and a mapping β_2 described by $\beta_2 : \mathcal{N}_1 \times \mathcal{N}_2 \times \mathcal{S} \rightarrow \hat{A}$. The decoder β_2 looks at the received codewords from the primary and secondary encoders and depending on its current state maps them into a reconstruction vector.

The modified FSVQ system described above will be referred to as FSVQ2 and it is the special case of what we will call NC-FSVQ2 for the noisy channel. Since the FSVQ2 system is a restricted version of the original FSVQ [18], we expect some performance degradation under noiseless channel conditions. However, as will be seen in the simulation results section, this degradation decreases as the encoding rate increases. On the other hand, when the channel is noisy, the structure of the FSVQ2 allows for a design of a joint source/channel code (NC-FSVQ2) and hence eliminates the need to transmit the "protected" state information as overhead and design a separate channel code. As a consequence, the resulting NC-FSVQ2 system does not have any additional inherent delay like NC-FSVQ1. Next we pose the problem of designing FSVQ2 in the presence of channel noise, develop necessary conditions for optimality and provide a design algorithm for NC-FSVQ2.

4.3.2 Noisy Channel FSVQ2 Problem Statement

Consider the block diagram in Fig. 4.3. The output of the primary encoder, u^1, is transmitted over a DMC described by a random mapping $\gamma_1 : \mathcal{N}_1 \rightarrow \mathcal{N}_1$ and the

transition matrix

$$Q_1(v^1|u^1) = Pr(\gamma_1(u^1) = v^1),\ \forall u^1, v^1 \in \mathcal{N}_1, \tag{4.25}$$

where v^1 is the output of the DMC γ_1. Similarly, the output of the secondary encoder, u^2, is transmitted over another DMC (assumed to be independent of γ_1) described by the random mapping $\gamma_2 : \mathcal{N}_2 \rightarrow \mathcal{N}_2$ and the transition matrix

$$Q_2(v^2|u^2) = Pr(\gamma_2(u^2) = v^2),\ \forall u^2, v^2 \in \mathcal{N}_2, \tag{4.26}$$

where v^2 is the output of the DMC γ_2. For the sake of simplicity of analysis, we are considering two different DMC's. In most practical situations, only one DMC is available; the outputs of the primary and secondary encoders are multiplexed and transmitted over this single DMC.

The decoder β_2 depends on the outputs of both channels v^1 and v^2 and is described by at most K^2 different codebooks $\mathcal{C}_{s,u^1} = \{\beta_2(u^1, u^2, s), u^2 \in \mathcal{N}_2\},\ \forall s \in \mathcal{S}, u^1 \in \mathcal{N}_1$. We will use $\mathcal{C}^{\text{sup}} = \{\mathcal{C}_{s,u^1}\}_{s\in\mathcal{S},u^1\in\mathcal{N}_1}$ to denote the collection of all the codebooks.

Our problem is to minimize the average distortion $D(\alpha_1, \alpha_2, \beta_2) = \frac{1}{L}E[d(\mathbf{X}, \mathbf{Y})]$ by appropriate design of α_1, α_2 and β_2 for given values of K and N_2 (which in turn determine the rate, R, of the system). Next, we develop necessary conditions for the optimality of the system.

4.3.3 Necessary Conditions

The average distortion is given by

$$\begin{aligned} D(\alpha_1, \alpha_2, \beta_2) &= \frac{1}{L}E[d(\mathbf{X}, \mathbf{Y})] \\ &= \frac{1}{L}\sum_{s,u^1,u^2}\sum_{\hat{s},v^1,v^2} E[d(\mathbf{X}, \mathbf{Y})|\hat{s}, v^1, v^2, s, u^1, u^2] \\ &\times\ Pr(\hat{S} = \hat{s}, V^1 = v^1, V^2 = v^2, S = s, U^1 = u^1, U^2 = u^2), \end{aligned} \tag{4.27}$$

where each uppercase letter denotes the random variable corresponding to the lowercase letter; s is the primary encoder state and $\hat{s}$ is the decoder state which, in general, can be different from each other due to the channel noise. Since

$$E[d(\mathbf{X},\mathbf{Y})|\hat{s},v^1,v^2,s,u^1,u^2] = \frac{\int_{\mathcal{P}^{u^2}_{s,u^1}} d(\mathbf{x},\beta_2(v^1,v^2,\hat{s}))p(\mathbf{x})d\mathbf{x}}{Pr(S=s,U^1=u^1,U^2=u^2)}, \tag{4.28}$$

we have

$$\begin{aligned} D(\alpha_1,\alpha_2,\beta_2) &= \frac{1}{L}\sum_{s,u^1,u^2}\int_{\mathcal{P}^{u^2}_{s,u^1}}[\sum_{\hat{s},v^1,v^2} Pr(V^2=v^2|U^2=u^2)Pr(\hat{S}=\hat{s}|S=s) \\ &\quad \times Pr(V^1=v^1|U^1=u^1)d(\mathbf{x},\beta_2(v^1,v^2,\hat{s}))]p(\mathbf{x})d\mathbf{x}. \end{aligned} \tag{4.29}$$

In obtaining (4.29), we have assumed that the DMC's γ_1 and γ_2 are independent of each other and that the encoder state at any time is just the codeword at output of the primary encoder transmitted at the previous time index and the decoder state is the corresponding received codeword. We denote the term in square brackets in (4.29) as a modified distortion measure

$$d_m(\mathbf{x},\beta_2(u^1,u^2,s)) = \sum_{\hat{s},v^1,v^2} Q_1(v^1|u^1)Q_1(\hat{s}|s)Q_2(v^2|u^2)d(\mathbf{x},\beta_2(v^1,v^2,\hat{s})). \tag{4.30}$$

Here, we have used the fact that u^1 (and therefore effectively s) is transmitted over DMC γ_1 and u^2 over γ_2. The modified distortion can be interpreted as the expected distortion in encoding $\mathbf{x}$ given that the primary encoder is in state s, and u^1 and u^2 are transmitted over DMC's γ_1 and γ_2; the average is taken with respect to the transition probabilities of the DMC's.

The average distortion can be expressed in terms of the modified distortion measure as

$$D(\alpha_1,\alpha_2,\beta_2) = \frac{1}{L}\sum_{s,u^1,u^2}\int_{\mathcal{P}^{u^2}_{s,u^1}} d_m(\mathbf{x},\beta_2(u^1,u^2,s))p(\mathbf{x})d\mathbf{x}. \tag{4.31}$$

Clearly, from the above expression, for a given $\mathcal{C}^{\text{sup}}$, the optimum partition $\mathcal{P}^{u^2}_{s,u^1}$

(hence optimum α_1, α_2) must satisfy

$$\mathcal{P}_{s,u^1}^{u^2*} = \{\mathbf{x} : d_m(\mathbf{x}, \beta_2(u^1, u^2, s)) \leq d_m(\mathbf{x}, \beta_2(u'^1, u'^2, s')), \forall s' \in \mathcal{S}, u'^1 \in \mathcal{N}_1, u'^2 \in \mathcal{N}_2\}. \tag{4.32}$$

This suggests that in order to find the optimum partition, we should encode the training sequence $\{\mathbf{x}_n\}$ using the Viterbi algorithm [53] (with an appropriate delay constraint) and the trellis associated with the FSVQ; the modified distortion measure is used as the branch metric [19]. The Viterbi algorithm will only be used in the design process, while in the encoding algorithm, the decision is made at each time index as we traverse along the trellis to avoid delay. Obviously, by introducing delay in encoding, improved performance can be expected [13].

For a given pair of encoders α_1 and α_2 (i.e., for a given partition $\{\mathcal{P}_{s,u^1}^{u^2}\}$), the optimum decoder β_2 should satisfy $\forall\, \hat{s}, v^1, v^2$

$$\beta_2^*(v^1, v^2, \hat{s}) = \arg \min_{\mathbf{y} \in \mathbb{R}^L} \sum_{s,u^1,u^2} Q_1(v^1|u^1) Q_1(\hat{s}|s) Q_2(v^2|u^2) \int_{\mathcal{P}_{s,u^1}^{u^2}} d(\mathbf{x}, \mathbf{y}) p(\mathbf{x}) d\mathbf{x}. \tag{4.33}$$

For the squared-error distortion measure, $d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||^2$, we have

$$d_m(\mathbf{x}, \beta_2(u^1, u^2, s)) = \sum_{\hat{s},v^1,v^2} Q_1(v^1|u^1) Q_1(\hat{s}|s) Q_2(v^2|u^2) ||\mathbf{x} - \beta_2(v^1, v^2, \hat{s})||^2. \tag{4.34}$$

Let $\forall s \in \mathcal{S}, u^1 \in \mathcal{N}_1, u^2 \in \mathcal{N}_2$

$$\mathbf{y}_{s,u^1,u^2} \triangleq \sum_{\hat{s},v^1,v^2} Q_1(v^1|u^1) Q_1(\hat{s}|s) Q_2(v^2|u^2) \beta_2(v^1, v^2, \hat{s}), \tag{4.35}$$

and

$$\delta_{s,u^1,u^2} \triangleq \sum_{\hat{s},v^1,v^2} Q_1(v^1|u^1) Q_1(\hat{s}|s) Q_2(v^2|u^2) ||\beta_2(v^1, v^2, \hat{s})||^2. \tag{4.36}$$

Then the modified distortion measure can be expressed as [28]

$$d_m(\mathbf{x}, \beta_2(u^1, u^2, s)) = ||\mathbf{x}||^2 - 2\langle \mathbf{x}, \mathbf{y}_{s,u^1,u^2} \rangle + \delta_{s,u^1,u^2}. \tag{4.37}$$

For given $\mathcal{C}^{\text{sup}}$, the terms $\mathbf{y}_{s,u^1,u^2}$ and δ_{s,u^1,u^2} can be precomputed and stored $\forall s \in \mathcal{S}, u^1 \in \mathcal{N}_1, u^2 \in \mathcal{N}_2$. Then to determine the optimum partition for a given $\mathcal{C}^{\text{sup}}$, the modified distortion can be computed using Equation (4.37).

For the squared-error distortion measure, it can be easily seen that the optimum decoder β_2 for a given partition $\{\mathcal{P}_{s,u^1}^{u^2}\}_{s,u^1,u^2}$ satisfies

$$\beta_2^*(v^1, v^2, \hat{s}) = E[\mathbf{X}|\hat{S} = \hat{s}, V^1 = v^1, V^2 = v^2] \tag{4.38}$$

$$= \frac{\sum_{s,u^1,u^2} Q_1(\hat{s}|s)Q_1(v^1|u^1)Q_2(v^2|u^2) \int_{\mathcal{P}_{s,u^1}^{u^2}} \mathbf{x}p(\mathbf{x})d\mathbf{x}}{\sum_{s,u^1,u^2} Q_1(\hat{s}|s)Q_1(v^1|u^1)Q_2(v^2|u^2) \int_{\mathcal{P}_{s,u^1}^{u^2}} p(\mathbf{x})d\mathbf{x}}. \tag{4.39}$$

The necessary conditions derived above assume that the source distribution is known. In practice, we use the training sequence approach and replace integrals by appropriate sums. Next we provide an algorithm based on these necessary conditions for designing a K-state NC-FSVQ2 of rate R bits/vector for a given training sequence $\{\mathbf{x}_n\}$.

4.3.4 Design Algorithm for NC-FSVQ2

0. Using the design algorithm described in Section 4.3.1 design an FSVQ2 with α_1 and α_2 as the primary and secondary encoders respectively, β_2 as the decoder and f' as the next-state function. Also denote the partition induced by α_1 and α_2 as $\{\mathcal{P}_{s,u^1}^{u^2}\}_{s,u^1,u^2}$. Let this FSVQ2 be an initial estimate of the NC-FSVQ2 to be designed by an iterative algorithm. In the subsequent steps, the next-state function is kept unchanged. The NC-FSVQ2 designed thus far works well only if the channel is noiseless. For the case when the channel cross-over probability is non-zero, the design algorithm proceeds as follows.

1. Given a small $\epsilon_0 > 0$ (stopping threshold), set $m = 0$ (m is the iteration index), $D^{(0)} = \infty$. Choose initial $\mathcal{C}^{\sup(0)}$ and $\{\mathcal{P}_{s,u^1}^{u^2(0)}\}_{s,u^1,u^2}$.

2. Set $m = m + 1$.

3. Compute $\mathbf{y}_{s,u^1,u^2}$ and δ_{s,u^1,u^2} as given by (4.35) and (4.36) using $\mathcal{C}^{\sup(m-1)}$.

4. Use the modified distortion measure given by (4.37) as the branch metric of the trellis corresponding to the current NC-FSVQ2; encode the training sequence using the Viterbi algorithm to obtain the optimal partition $\{\mathcal{P}_{s,u^1}^{u^2*}\}_{s,u^1,u^2}$ and define

$$\alpha_1^*(\mathbf{x}, s) = u^1, \text{ if } \mathbf{x} \in \mathcal{P}_{s,u^1}^*, \tag{4.40}$$

$$\alpha_2^*(\mathbf{x}, s, u^1) = u^2, \text{ if } \mathbf{x} \in \mathcal{P}_{s,u^1}^{u^2*}, \tag{4.41}$$

where $\mathcal{P}_{s,u^1}^* = \bigcup_{u^2} \mathcal{P}_{s,u^1}^{u^2*}$.

5. Compute the average distortion $D^{(m)}$ using $\mathcal{C}^{\text{sup}(m-1)}$, the optimum partition $\{\mathcal{P}_{s,u^1}^{u^2*}\}_{s,u^1,u^2}$ obtained in step (4) and Equations (4.31) and (4.37). Compute the optimum codebook $\mathcal{C}^{\text{sup}*}$ using $\{\mathcal{P}_{s,u^1}^{u^2*}\}_{s,u^1,u^2}$ and (4.39). Set $\mathcal{C}^{\text{sup}(m)} = \mathcal{C}^{\text{sup}*}$, $\mathcal{P}_{s,u^1}^{u^2(m)} = \mathcal{P}_{s,u^1}^{u^2*}$, $\alpha_1^{(m)} = \alpha_1^*$ and $\alpha_2^{(m)} = \alpha_2^*$.

6. If $\frac{D^{(m-1)} - D^{(m)}}{D^{(m)}} > \epsilon_0$, go to step (2).

7. Stop with $\{\mathcal{P}_{s,u^1}^{u^2(m)}\}_{s,u^1,u^2}$ and $\mathcal{C}^{\text{sup}(m)}$ as the final partition and codebook respectively of the NC-FSVQ2. The final primary and secondary encoders of the NC-FSVQ2 are $\alpha_1^{(m)}$ and $\alpha_2^{(m)}$, respectively.

Note that at each step of iteration the average distortion $D^{(m)}$ can only decrease and since it is lower bounded by zero, the sequence $\{D^{(m)}\}$ has to converge at least to a local minimum.

4.4 Simulation Results

In all the experiments, we have considered a binary symmetric channel (BSC) with bit error rates (BERs) $\epsilon = 0.0, 0.005, 0.01, 0.05, 0.1$. Performances of NC-FSVQ1 (FSVQ1 when $\epsilon = 0.0$) and NC-FSVQ2 (FSVQ2 when $\epsilon = 0.0$) are presented in this section for the Gauss-Markov (G-M) source and the speech LSP parameters.

First, we provide the results for the G-M source under noiseless as well as noisy channel conditions and then consider the LSP parameters.

4.4.1 Gauss-Markov Source

We used 200,000 vectors of dimension 4 from the G-M source with $\rho = 0.9$ as the training sequence. The test sequence consisted of 100,000 4-dimensional vectors from the same source not included in the training sequence. The distortion measure used was squared-error; the performance results are reported in signal-to-noise ratio (SNR) in dB.

Performance of FSVQ1

8-state FSVQ1 was designed for various bit rates (denoted by b) in bits/vector. The results are tabulated in Table 4.1. For comparison, the results obtained using LBG-VQ are also included. Table 4.1 indicates that FSVQ1 outperforms LBG-VQ at all rates by at least 1 dB. These results are in agreement with those of [18].

Noisy Channel Performance of NC-FSVQ1

FSVQ1 was simulated in the presence of channel noise and the results are presented in Table 4.2. Severe degradation in the performance of FSVQ1 is observed and the SNR actually often decreases with increase in b.

The performance of NC-FSVQ1 along with the results for CO-VQ are displayed in Fig. 4.4 in order to make comparisons. Additional results on NC-FSVQ2 are also included in Fig. 4.4 and will be discussed later. When $0.005 \leq \epsilon \leq 0.01$ ($0.05 \leq \epsilon \leq 0.1$), a rate 1/2 (1/3) convolutional code was used for protecting the state index and $p = 6$ ($p = 3$). The bit rates in Fig. 4.4 include the overhead information associated with the transmission of protected encoder state information. When an 8-state NC-FSVQ1 is used, the overhead is given by $3/(pr_c)$, bits/vector, where

r_c is the rate of the channel code. Close study of Fig. 4.4 shows that NC-FSVQ1 and CO-VQ perform very close for all values of b when $\epsilon \leq 0.05$. At $\epsilon = 0.1$, NC-FSVQ1 outperforms CO-VQ by 0.4-0.9 dB. However, as compared to the FSVQ of [18], NC-FSVQ1 performs significantly better for all $\epsilon > 0.0$.

Performance of FSVQ2

8-state FSVQ2 (NC-FSVQ2 for $\epsilon = 0.0$) was designed using the algorithm described in Section 4.3.1. Table 4.3 includes the results for FSVQ2 and LBG-VQ for different values of b. FSVQ2 outperforms LBG-VQ by 0.56-0.85 dB for different values of b. As compared to the original FSVQ [18], the performance of FSVQ2 is inferior by about 0.5 dB due the imposed structure in the system. FSVQ2 was also designed with 16 states but it gave very marginal improvement over the 8-state FSVQ2.

Noisy Channel Performance of NC-FSVQ2

NC-FSVQ2 was designed for different values of b and ϵ using the algorithm described in Section 4.3.4. The performance of the 8-state NC-FSVQ2 is included in Fig. 4.4 which shows that NC-FSVQ2 outperforms the CO-VQ for all values of b and ϵ. At $\epsilon = 0.005$, the SNR gain over CO-VQ at the same bit rate is about 0.4-0.8 dB. As ϵ increases, the gain increases; at $\epsilon = 0.1$, the gain is about 0.7-1.0 dB. NC-FSVQ2 outperforms NC-FSVQ1 for $\epsilon \leq 0.05$, however when $\epsilon = 0.1$, the two schemes are comparable in terms of SNR. In terms of computational complexity, NC-FSVQ2 has the complexity of a two-stage VQ, while NC-FSVQ1 (operating at the same rate) has the much higher complexity of a full-searched VQ; in addition, there is a delay involved in the NC-FSVQ1 at the decoder.

4.4.2 Speech LSP Parameters

The speech database used for training consisted of 120 minutes of speech (from the TIMIT database) sampled at 8 KHz and uttered by several male and female speakers. A 10th-order LPC analysis based on the standard autocorrelation method was performed every 22.5 msec with a 30 msec analysis window. Thus we had 320,000 LSP vectors in the training sequence. The test sequence consisted of 2,261 vectors not in the training sequence (also used in [54]).

The speech LSP parameters exhibit relatively high correlation within a frames well as between adjacent frames and are therefore good candidates for FSVQ type systems that can efficiently exploit these correlations. The final performance of the LSP quantization systems is expressed in terms of the average spectral distortion (SD) given by

$$\mathrm{SD} = \frac{1}{T}\sum_{n=1}^{T}[\int_{-\pi}^{\pi}(10\log S_n(\omega) - 10\log \hat{S}_n(\omega))^2\frac{d\omega}{2\pi}]^{\frac{1}{2}},\ (\mathrm{dB}). \qquad (4.42)$$

Here, $S_n(\omega)$ and $\hat{S}_n(\omega)$ are the original and quantized spectra, respectively, associated with the n^{th} frame and T is the number of frames. The average spectral distortion is a useful measure of performance in LSP quantization schemes [34]. Earlier work on memoryless VQ of LSP parameters [33] suggests that for transparent[4] quantization of LSP parameters, an encoder rate in the neighborhood of 24 bits/frame is required. To achieve this kind of rate, FSVQ state codebooks will need to have 2^{24} codevectors which is prohibitively large. Thus, to reduce the complexity, we split each 10-dimensional LSP vector into 3 subvectors (dimensions 3, 3 and 4) and design an FSVQ of rate $R/3$ for each subvector (R is the overall rate in bits/frame); the three subvectors are encoded separately using different FSVQs[5].

[4]The quantization is said to be transparent if the resulting average spectral distortion [34] is less than 1.0 dB and the fraction of frames with spectral distortion over 2.0 dB (called the outliers) is less than 2% with no frame having spectral distortion greater than 4.0 dB.

[5]In [33], the LSP vectors are split into 2 subvectors and a 1 dB average spectral distortion is

Performance results of such a system are provided next and it is shown that noticeable gains are obtained over the case where a memoryless LBG-VQ is used to encode each subvector (such a system will be called a split-VQ system).

Performance of FSVQ1

The FSVQ1 was designed for each subvector resulting in a split-FSVQ1 system. The squared-error distortion measure was used for the design while encoding was done using the IHM distortion measure introduced in [54]. Table 4.4 shows the results of split-FSVQ1 (8 states for each subvector) and split-VQ (using the same 3 subvectors) at various bit rates (again denoted by b). These results indicate that split-FSVQ1 yields a saving of approximately 3 bits/frame over the split-VQ system and achieves the 1 dB average spectral distortion (SD) at 24 bits/frame; at this rate the outlier rate (OL) is 0.75%. Performance of split-FSVQ1 can be improved by increasing the number of states; we found that a 16-state FSVQ1 (for each subvector) achieves transparent quantization at 23 bits/frame.

Noisy channel performance of NC-FSVQ1

The split-FSVQ1 system was simulated in the presence of channel noise and the results are tabulated in Table 4.5. Clearly, the performance degradation is severe. With 24 bits/frame, under noiseless conditions, the split-FSVQ1 achieves 1 dB average SD but when $\epsilon = 0.01$, the average SD rises up to 2.97 dB.

The performance results of NC-FSVQ1[6] (designed for $\epsilon > 0.0$) are given in Fig. 4.5 for $p = 4$ and $p = 8$. To protect the encoder state information, we found

reported with a 12-bit VQ for each subvector. We could not, however, repeat this result with 2 subvectors. We suspect that one of the reasons for this is the use of the modified covariance LPC analysis (as opposed to the standard autocorrelation method used here) in [33]. An additional reason may be that [33] considers the telephone band (200-3300 Hz) frequancies for computing SD as opposed to the full-band (0-4000 Hz) used in this thesis.

[6]As in the noiseless case, we are using a different NC-FSVQ1 for each subvector, but for brevity we use the terminology NC-FSVQ1 instead of split-NC-FSVQ1.

that a rate 1/2 (1/3) convolutional code with constraint length 4 is sufficient when $0.005 \leq \epsilon \leq 0.01$ ($0.05 \leq \epsilon \leq 0.1$). The decoder is implemented using the Viterbi algorithm and the estimated codeword is released at the end of each frame. The bit rates in Fig. 4.5 also include the overhead (for NC-FSVQ1) associated with the transmission of the protected encoder state sequence. When 3 subvectors and 8-state FSVQs are used, the overhead is easily computed to be $(3 \times 3)/(pr_c)$, bits/frame, where r_c is the rate of the channel code used.

For comparisons, the results for the split CO-VQ system (each subvector is quantized using a CO-VQ) are also included in Fig. 4.5. Study of Fig. 4.5 shows that, with a delay of 4 frames, when $\epsilon = 0.005$, NC-FSVQ1 performs close to split CO-VQ but as ϵ increases, NC-FSVQ1 starts performing better than CO-VQ with a saving of about 1.5 bits/frame at $\epsilon = 0.01$; when $\epsilon = 0.1$, the saving is more than 4.25 bits. If we can tolerate a delay of 8 frames, the gains are even higher; at $\epsilon = 0.005$, the gain is about 1.75 bits/frame, while at $\epsilon = 0.1$, the gain is over 5 bits/frame. Also comparison with the MSVQ-based scheme of [28], [54] (see Fig. 4.5) shows significant gains at larger values of ϵ (> 0.01) and comparable or better performance at $\epsilon = 0.005$. When $\epsilon = 0.005$, NC-FSVQ1 achieves a 1 dB average SD with 31.5 bits/frame ($p = 4$) and 29.25 bits/frame ($p = 8$). We believe (from [33]) that the 1 dB average SD can be obtained at lower rates if we use the modified covariance method of analysis [35] and weighted distortion measure of [33] and compute the SD in the 200-3300 Hz frequency band.

Performance of FSVQ2

The FSVQ2 described in Section 4.3, was designed using the algorithm described in Section 4.3.1 for each subvector resulting in split-FSVQ2 system. Again, squared-error distortion was used for the design, while encoding was done using the IHM distortion measure. Table 4.6 shows the results of the 8-state split-FSVQ2 and

split-VQ for various values of b. The results show that as compared to the split-VQ system, split-FSVQ2 achieves a saving of approximately 2 bits/frame (1 bit/frame less than split-FSVQ1 for higher values of b). Split-FSVQ2 achieves close to 1 dB average SD at 25 bits/frame; the value of OL at this rate is 1.1%. The performance of split-FSVQ2 improves as the number of states increases and a 16-state split-FSVQ2 achieves transparent quantization at 24 bits/frame.

Noisy channel performance of NC-FSVQ2

The performance results of NC-FSVQ2[7] (for $\epsilon > 0.0$) are given in Fig. 4.6 for the 8- and 16-state case. NC-FSVQ2 is designed for each of the three subvectors using the design algorithm of Section 4.3.4. For comparison purposes, the results of split CO-VQ are also plotted in Fig. 4.6. A close study of the figure shows that 8-state NC-FSVQ2 outperforms split CO-VQ by over 1 bit/frame when $\epsilon = 0.005$; this gain increases to over 3 bits/frame at $\epsilon = 0.1$. Comparison of the 8-state and 16-state NC-FSVQ2 shows that unlike in the case of G-M source, the performance improvement with the increase in the number of states is noticeable for higher values of ϵ for speech LSP parameters. When $\epsilon = 0.05$, the improvement in going from 8 to 16 states is about 0.5 bits/frame, while for $\epsilon = 0.1$, the gain can be as high as 1.5 bits/frame. The gain is insignificant for $\epsilon \leq 0.01$.

NC-FSVQ1 versus NC-FSVQ2

Comparison of Figs. 4.5 and 4.6 shows that at low values of ϵ (< 0.01), NC-FSVQ2 performs better than NC-FSVQ1 and vice versa for high values of ϵ (> 0.05), with comparable results for $0.01 \leq \epsilon \leq 0.05$. However, NC-FSVQ2 does not suffer from any delay. In addition, the decoder for NC-FSVQ2 is simpler and there is no need for a separate channel code in NC-FSVQ2. The memory requirements are the same

[7]Again, as in the noiseless case, we are using a different NC-FSVQ2 for each subvector, but for brevity we use the terminology NC-FSVQ2 instead of split-NC-FSVQ2.

for NC-FSVQ1 and NC-FSVQ2 operating at the same bit-rate and with the same number of states.

4.5 Summary and Conclusions

In this chapter, we have considered the quantization of sources with memory using FSVQ type and LBG-VQ systems under noiseless as well as noisy channel conditions. Two sources, namely the G-M source with $\rho = 0.9$ and speech LSP parameters were considered. Under noiseless channel conditions, comparisons were made between FSVQ1, FSVQ2 and LBG-VQ systems for both sources. It was concluded that in all cases both FSVQ systems outperform LBG-VQ with FSVQ1 performing slightly better than FSVQ2.

Under noisy channel conditions, FSVQ1 collapsed for all values of $\epsilon > 0.0$, while FSVQ2 collapsed for $\epsilon > 0.01$ making it necessary to redesign these FSVQ systems by taking into account the channel noise. Two systems namely NC-FSVQ1 (noisy version of FSVQ1) and NC-FSVQ2 (noisy version of FSVQ2) were described based on developments in joint source-channel trellis coding systems and again used to quantize the G-M source and speech LSP parameters. For both sources, the noisy channel FSVQ systems offered significantly higher robustness compared to their noiseless channel counterparts. In particular, when LSP parameters were quantized using NC-FSVQ1, the saving over split CO-VQ was 5 bits/frame for very noisy channels with a decoding delay of 8 frames; NC-FSVQ2 achieved a saving of up to 3 bits/frame (for the 8-state case) over split CO-VQ with no delay.

Based on the results summarized above, it appears that noisy channel FSVQs are good candidates for speech encoding systems such as in mobile communication where channel noise is an important factor. If a delay can be tolerated, then NC-FSVQ1 is a better choice especially for the highly noisy case (ϵ close to 0.1) and when delay is an issue then NC-FSVQ2 is a better candidate for all levels of

channel noise.

Also if delay is not an issue, then introducing encoding delay in NC-FSVQ1 and NC-FSVQ2 will lead to better performance (the system will become a trellis encoding system with a general next-state map); FSVQ with a delay in the encoder was considered in [13] and performance improvements were observed over the ordinary FSVQ for the noiseless channel case.

In the NC-FSVQ1 and NC-FSVQ2 schemes, each of the three subvectors of each frame are treated independently; we can expect performance improvements if we exploit the memory present between the three subvectors. Also, if we split the LSP vectors into just two subvectors, then the overhead needed in the case of NC-FSVQ1 will be reduced; in addition, the intraframe correlation will be better utilized than in the 3 split case. It is also expected that all our results will be further improved by about 2 bits/vector if we use the modified covariance method for analysis and the weighted squared-error for the design and encoding [33] and compute the average SD over the band 200-3300 Hz.

b	LBG-VQ	FSVQ1
1	3.57	5.26
2	6.53	8.14
3	8.24	9.97
4	10.15	11.31
5	11.60	12.68
6	12.98	14.04
7	14.31	15.34

Table 4.1: Performance of LBG-VQ and FSVQ1 under noiseless channel conditions at various bits rates (bits/vector); Gauss-Markov source with $\rho = 0.9$; $L = 4$; $K = 8$.

b	$\epsilon = 0.000$	$\epsilon = 0.005$	$\epsilon = 0.010$	$\epsilon = 0.050$	$\epsilon = 0.100$
1	5.26	4.21	3.30	0.40	-0.87
2	8.14	4.65	2.99	-0.79	-1.84
3	9.97	5.40	3.38	-0.93	-2.07
4	11.31	4.30	1.98	-1.67	-2.53
5	12.68	4.99	2.66	-1.50	-2.48
6	14.04	4.78	2.36	-1.74	-2.62
7	15.34	3.74	1.54	-2.27	-2.94

Table 4.2: Performance of FSVQ1, for various levels of channel noise and different bit rates; Gauss-Markov source with $\rho = 0.9$; $L = 4$; $K = 8$.

b	LBG-VQ	FSVQ2
3	8.24	9.07
4	10.15	10.84
5	11.60	12.36
6	12.98	13.63
7	14.31	14.92
8	15.69	16.25

Table 4.3: Performance of LBG-VQ and FSVQ2 under noiseless channel conditions at various bits rates (bits/vector); Gauss-Markov source with $\rho = 0.9$; $L = 4$; $K = 8$.

	Split-VQ		Split-FSVQ1	
b	SD	OL%	SD	OL%
21	1.51	9.69	1.27	3.80
22	1.43	6.59	1.17	2.60
23	1.33	3.80	1.09	1.60
24	1.25	1.95	1.02	0.75
25	1.18	1.37	0.97	0.53
26	1.11	0.57	0.91	0.44
27	1.03	0.18	0.86	0.35

Table 4.4: Performance of split-VQ and -FSVQ1 under noiseless channel conditions at various bit rates (bits/frame); speech LSP parameters; $K = 8$.

	$\epsilon = 0.005$	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$
b	SD	SD	SD	SD
21	2.36	3.23	6.50	7.66
22	2.24	3.08	6.38	7.81
23	2.19	3.13	6.46	7.85
24	1.90	2.97	6.40	7.66
25	1.77	2.86	6.26	7.60
26	1.80	2.87	6.11	7.52
27	1.79	2.70	6.03	7.41

Table 4.5: Performance of split-FSVQ1 for various levels of channel noise and bit rates; speech LSP parameters; $K = 8$.

	Split-VQ		Split-FSVQ2	
b	SD	OL%	SD	OL%
21	1.51	9.69	1.39	6.28
22	1.43	6.59	1.30	4.80
23	1.33	3.80	1.20	2.73
24	1.25	1.95	1.10	1.78
25	1.18	1.37	1.02	1.10
26	1.11	0.57	0.95	0.70
27	1.03	0.18	0.89	0.51

Table 4.6: Performance of split-VQ and -FSVQ2 under noiseless channel conditions at various bit rates (bits/frame); speech LSP parameters; $K = 8$.

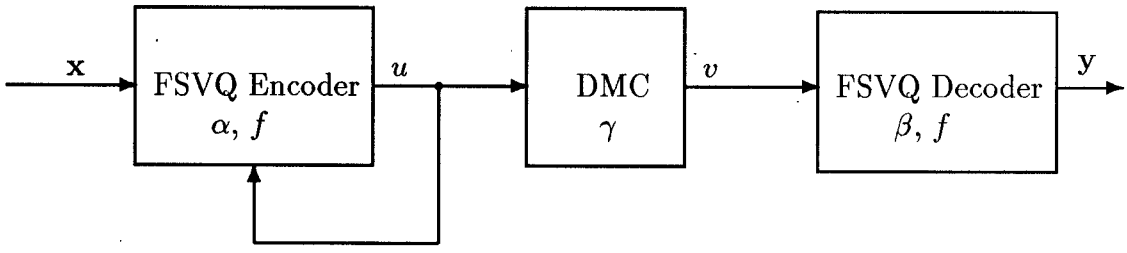
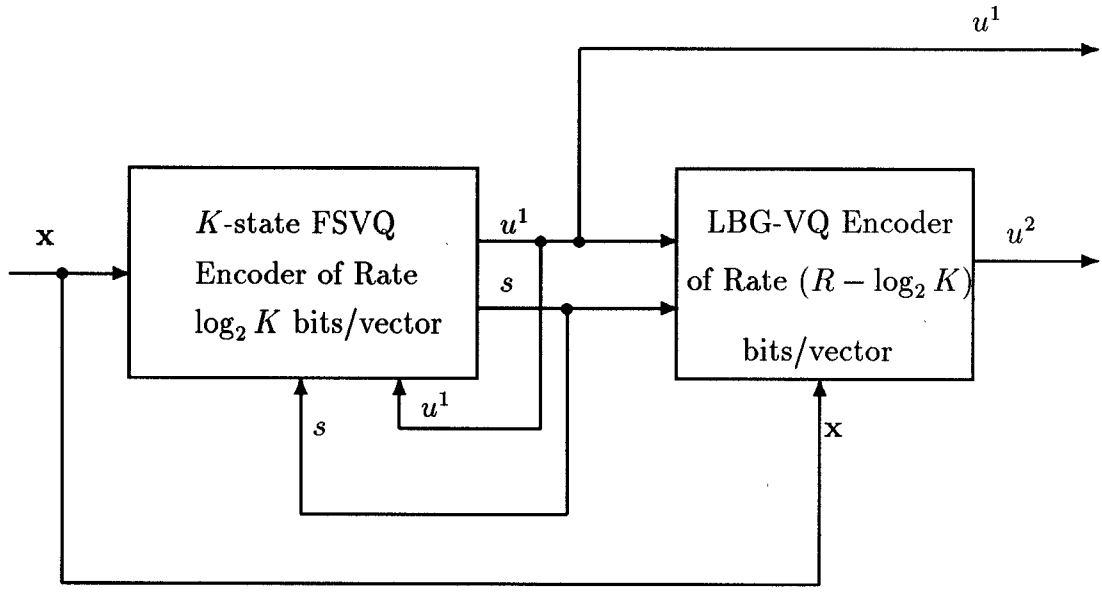
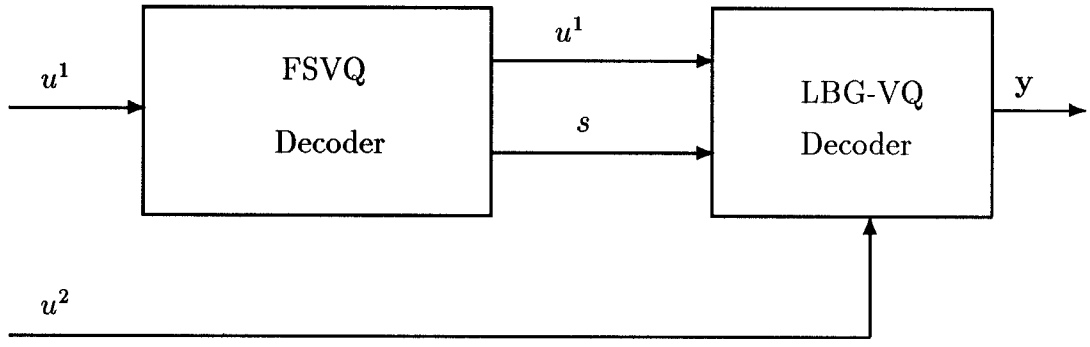


Figure 4.1: Block diagram of an FSVQ system with transmission over a noisy channel.



BLOCK ENCODER



BLOCK DECODER

Figure 4.2: Block diagram of the encoder and the decoder of the FSVQ2 system.

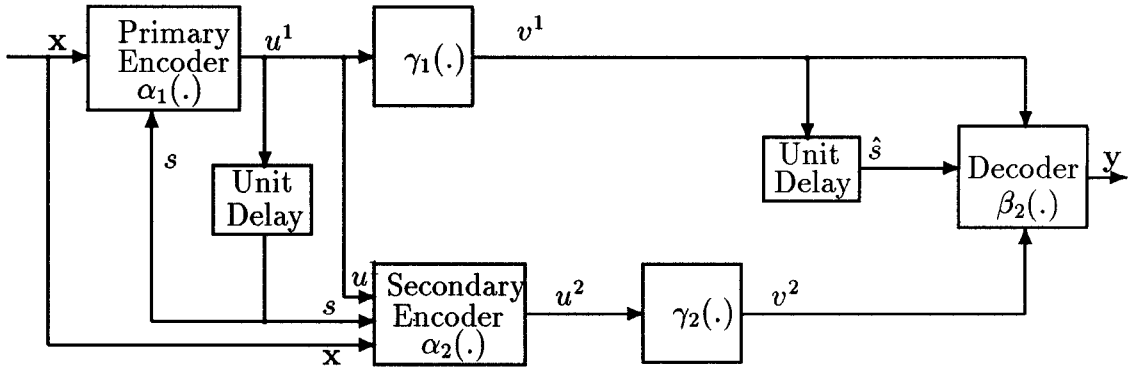


Figure 4.3: Block diagram of NC-FSVQ2 system with transmission over a noisy channel.

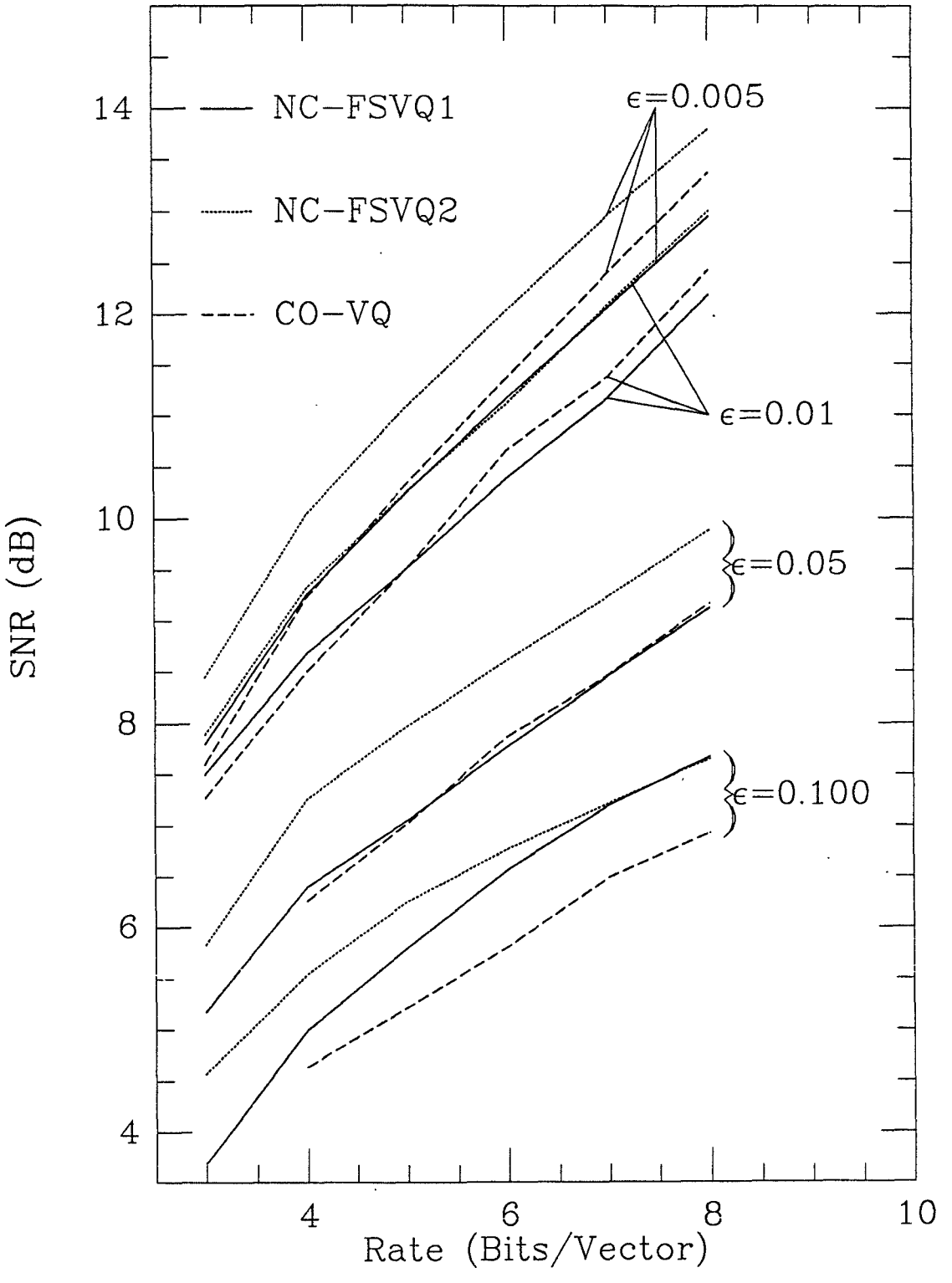


Figure 4.4: Performance of NC-FSVQ1, NC-FSVQ2 and CO-VQ for various levels of channel noise and bit rates; $p = 6$, when $\epsilon = 0.005$ and 0.010, while $p = 3$, when $\epsilon = 0.050$ and 0.100; Gauss-Markov source with $\rho = 0.9$; $L = 4$; $K = 8$.

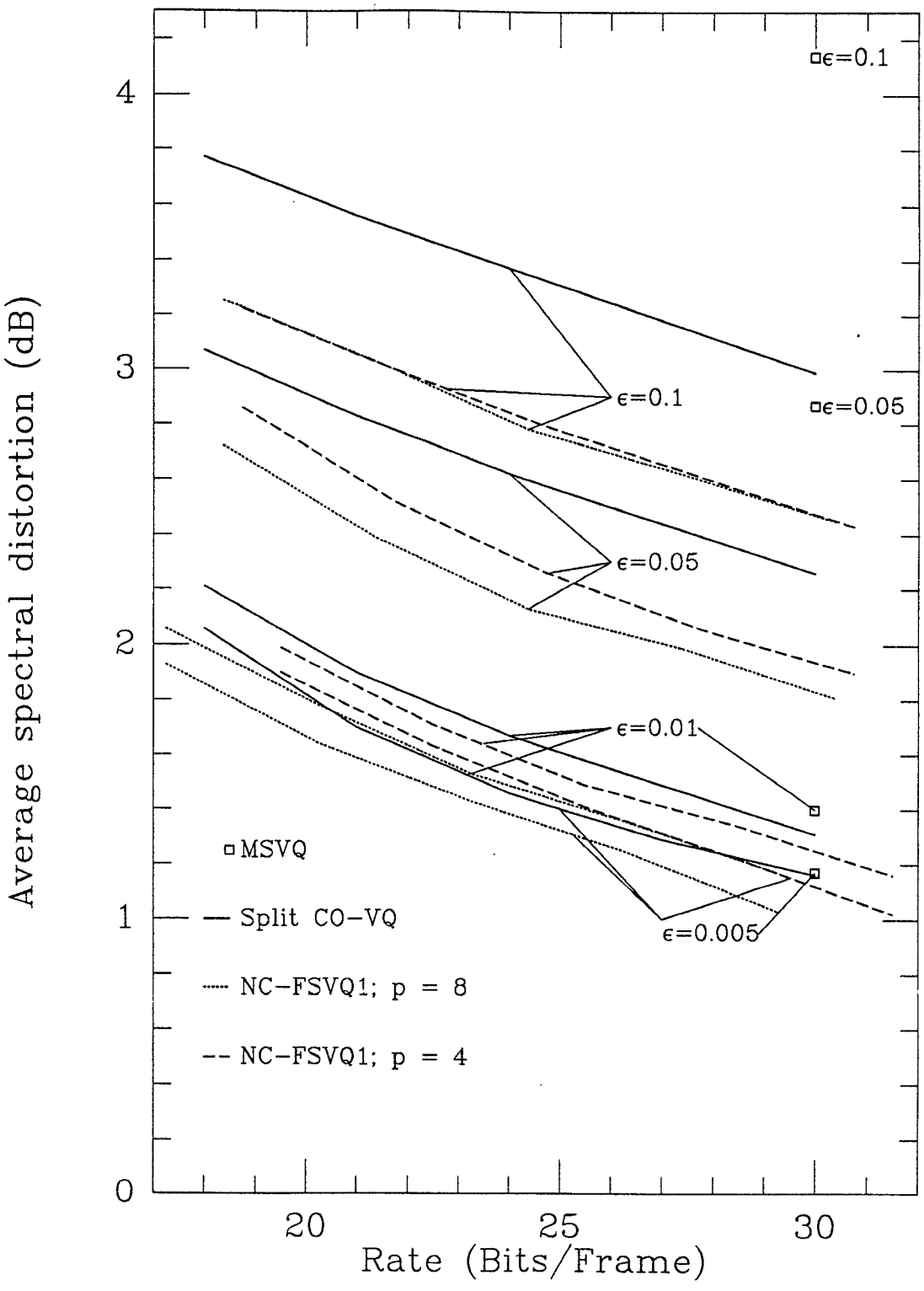


Figure 4.5: Performance of NC-FSVQ1 and split CO-VQ for various levels of channel noise and bit rates; speech LSP parameters; $K = 8$; $p = 4$ and $p = 8$; MSVQ results are taken from [54].

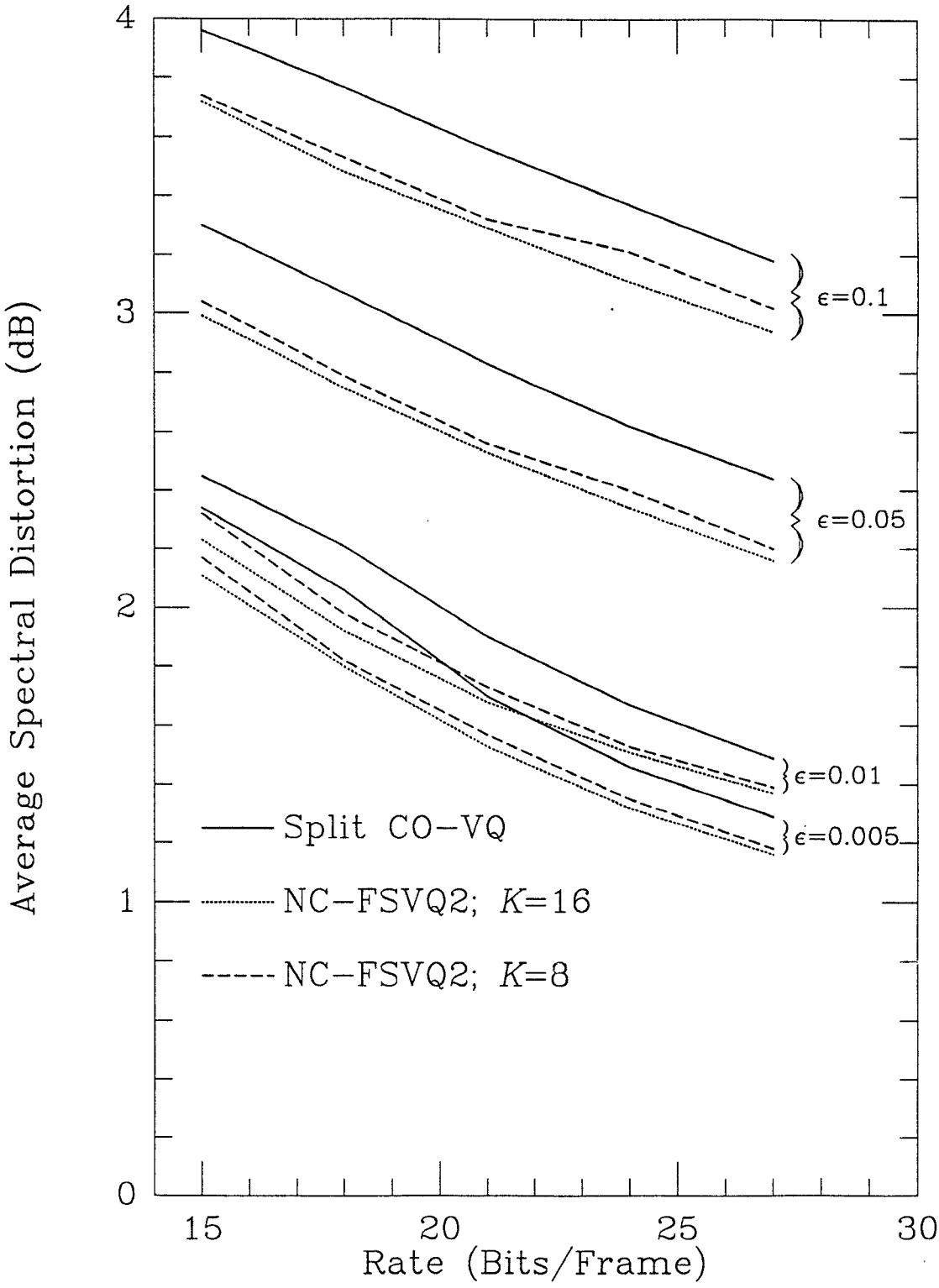


Figure 4.6: Performance of NC-FSVQ2 and split CO-VQ for various levels of channel noise and bit rates; speech LSP parameters; $K = 8$ and $K = 16$.

Chapter 5

Conclusions and Suggestions for Future Work

In Chapter 2, we have considered two *variable-rate* variations of the FSVQ system described in [18]. In addition, we have considered the possibility of using structured VQs as possible candidates for the state quantizers of the FSVQ of [18]. The design algorithms for the variable-rate and other versions of FSVQs are obtained by simple modification of the FSVQ design algorithm. None of these algorithms (including the one described in [18]) converge and there is no guaranteed improvement at each step of iteration. The variable-rate FSVQ systems considered in Chapter 2 perform substantially better than the system in [18] and SQNR gains of up to 4.25 dB on in-training sequence and 3 dB on out-of-training test sequences are obtained for encoding a 1-D source like the sampled speech waveform. We believe that higher gains can be achieved even on the test sequence outside the training sequence if a sufficiently long sequence is used to train the variable-rate FSVQ systems. Similar results were obtained for the composite Gauss-Markov source. One shortcoming of the variable-rate FSVQ systems is that if it is designed to achieve a certain encoding rate for a training sequence, it may give a different rate if used to encode another sequence. To circumvent this difficulty, some type of algorithm which adaptively modifies the encoding system based on some local (instantaneous) measurement of the encoding rate is needed.

In chapter 3, we considered several variations of two low bit rate image coding systems based on 2-D fixed-rate and variable-rate FSVQs. The design algorithms for various 2-D FSVQs were obtained by extensions of the algorithms given for the 1-D case. Out of all the image coding schemes considered in Chapter 3, PR-VFS-UTSVQs have the potential of doing the best. They achieve a PSNR of 30.74 at 0.25 bits/pixel and the reconstructed image quality is good. Although, the PSNR value achieved at $b = 0.31$ bits/pixel was 31.2 dB for encoding the 512×512 version of "Lena", PR-VFS-UTSVQs have the potential to do better; we believe that the gain in the PSNR value at $b = 0.31$ bits/pixel over that at $b = 0.25$ bits/pixel can be higher if we use a larger training sequence.

While the performance of various FSVQs was observed to be consistently better than the memoryless LBG-VQ under noiseless channel conditions (such conditions were assumed throughout Chapters 2 and 3), their performance degraded significantly in the presence of channel noise and it became necessary to redesign the FSVQ systems taking into account the channel noise. Under noisy channel conditions, we restricted to the fixed-rate FSVQ system with LBG-VQ as the state quantizers. Two systems namely NC-FSVQ1 and NC-FSVQ2 were described in Chapter 4 based on developments in joint source-channel trellis coding systems and used to quantize the G-M source and speech LSP parameters. For both sources, the noisy channel FSVQ systems offer significantly higher robustness compared to their noiseless channel counterparts. In particular, when LSP parameters were quantized using NC-FSVQ1, the saving over split CO-VQ was 5 bits/frame for very noisy channels with a decoding delay of 8 frames; NC-FSVQ2 achieved a saving of up to 3 bits/frame (for the 8-state case) over split CO-VQ with no delay.

5.1 Suggestions for Future Research

There are several directions in which this work can proceed. As a first step, a delay can be introduced in the encoder of various FSVQs described in Chapters 2 and 3. Incorporation of delay enables the encoder to make a decision that will be good in the long term sense leading to a better overall performance. For encoding still images, delay of up to one complete frame is tolerable in some applications and therefore we can further improve the performances of ME-VFS-UTSVQs and PR-VFS-UTSVQs by using a "lookahead" kind of encoder for encoding images. We also expect the FSVQ of [18] with delay to perform better than vector trellis encoding system [12] because of the use of a more general next-state function as opposed to a shift-register based implementation of the next-state map.

Another avenue is to apply variable-rate FSVQ systems for video compression. We can extend the 2-D FSVQs described in Chapter 3 to three dimensions. The state variable for the 3-D case has an additional component that essentially exploits the time correlation. In addition, we can use motion compensation. The resulting 3-D variable-rate FSVQs will exploit intra-frame and inter-frame correlation very efficiently leading to a high compression ratio.

It has been demonstrated in Chapter 2 of this thesis that in an FSVQ, the state quantizer can be any type of VQ. The choices of state VQs considered in this dissertation are limited to LBG-VQ, TSVQ and UTSVQ. The other choices are MSVQ and an LBG-VQ followed by a scalar quantizer; with these two choices very high rate FSVQs can be designed. For encoding LSP parameters in Chapter 4, FSVQs had to be designed with rates as high as 27 bits/frame. In order to achieve such high rates, each LSP vector was split into 3 subvectors which were treated independently of each other and as a consequence of this, the intra-frame correlation was not fully exploited. An alternative that can exploit intra-frame

correlation better (and perhaps lead to better performance than split-FSVQ1) would be to use MSVQ or 'an LBG-VQ followed by scalar quantizers', with a first-stage VQ of large rate in order to exploit most of the intra-frame correlation; inter-frame correlation will of course be exploited by the finite-state structure of FSVQ.

The FSVQ proposed by Foster has a feedback structure; the current output and the current state determine the next encoder/decoder state. It is also possible to consider a system in which the encoder state is decided based on the current input (as opposed to the previous output). Such a system is called forward adaptive system [17] and since the encoder state is derived from the input, the decoder in forward adaptive system cannot determine the encoder state without receiving extra information from the encoder. However, for correlated sources like speech and image, the state sequence tends to be highly correlated and also have a very skewed distribution. Assuming that the state sequence can be modeled by a Markov source and that the state sequence is entropy-coded, forward adaptive versions of variable-rate FSVQs can be designed. Some simulations were performed on sampled speech source under the above assumptions and an improvement of over 1 dB was observed over the best case results of VFS-UTSVQ using a forward adaptive version of VFS-UTSVQ.

An interesting and a nontrivial problem arises as a result of letting the cardinalities of the codebooks associated with different states to be different. A variable-rate FSVQ system designed to achieve a certain average rate for a training sequence, when used to encode another sequence may result in an average rate off by a substantial amount from the designed average rate, which is undesirable. To circumvent this difficulty, the method that we propose is to use the idea of recently developed finite-state structured vector quantizer [41], [42] to convert the variable-rate FSVQ systems into a fixed-rate system. By incorporating some delay into

the system, the codebook search algorithm of [41] can be used to encode several vectors (say n) at a time to give a fixed-rate system performing very close to the original variable-rate system.

Another interesting problem would be to design variable-rate forward adaptive systems whose codewords would be transmitted over a noisy channel. Unequal channel error protection would be a natural requirement here (especially for UTSVQ based system where higher protection is needed for nodes with higher goodness of split). A good candidate for the error control code would be the rate-compatible punctured convolutional (RCPC) codes [55]. In fact Tanabe and Farvardin have successfully used the RCPC codes in a system that uses entropy coding of image subbands [56].

In Chapter 4, two noisy channel FSVQ systems were proposed that resulted out of an attempt to simplify the FSVQ structure so that a certain probability function could be evaluated easily. In NC-FSVQ1, it was assumed that the "protected" encoder state is transmitted to the receiver at all times, while in NC-FSVQ2 some conditions were imposed on the next-state function of the FSVQ. NC-FSVQ2 was found to be more attractive since unlike NC-FSVQ1, it did not have any decoding delay. However, due to the imposition of structure on NC-FSVQ2, its performance suffered degradation as compared to FSVQ of [18] under noiseless channel conditions. It appears that performance better than NC-FSVQ2 should be achievable if the structure of FSVQ is kept unchanged and the desired probability function is somehow computed directly (In [57], the required probability is computed for a balanced trellis and probably the method used therein can be used as guideline).

Appendix A

Codeword Reassignment Algorithm for NC-FSVQ1

Assume that the encoder and decoder are in the same state at time n, say, s_n. Let s_{n+1} and $\hat{s}_{n+1}$ denote the encoder and the decoder state, respectively, at time $n+1$. Then, $Pr(s_{n+1} = i, \hat{s}_{n+1} = j|s_n)$ is the probability that the state at time $n+1$ is incorrectly decoded as j instead of i given that both encoder and decoder were in state s_n at time n. This probability can be expressed as

$$Pr(s_{n+1} = i, \hat{s}_{n+1} = j|s_n) = Pr(s_{n+1} = i|s_n)Pr(\hat{s}_{n+1} = j|s_{n+1} = i, s_n), \quad \text{(A.1)}$$

where

$$Pr(s_{n+1} = i|s_n) = \sum_{u_n : f(u_n, s_n) = s_{n+1}} Pr(u_n|s_n), \quad \text{(A.2)}$$

and

$$Pr(\hat{s}_{n+1} = j|s_{n+1} = i, s_n) = \sum_{v_n : f(v_n, s_n) = j} \sum_{u_n : f(u_n, s_n) = i} Q(v_n|u_n). \quad \text{(A.3)}$$

Also

$$Pr(s_{n+1} = i, \hat{s}_{n+1} = j) = \sum_{s_n} Pr(s_n)Pr(s_{n+1} = i, \hat{s}_{n+1} = j|s_n). \quad \text{(A.4)}$$

The above expression gives the probability of confusing state i with state j at any time given that the encoder and decoder states were the same at the previous time; we say that state i is confused as state j and denote the probability of this event

by $P_{i,j}$, $\forall i,j \in \mathcal{S}$ which can be computed using Equations (A.1)-(A.4).

Once we compute the probabilities $P_{i,j}$, we find the pair I, J ($I \neq J$) such that $P_{I,J} \geq P_{i,j}$, $\forall i,j \in \mathcal{S}$ and then we rearrange the codevectors (reassign codeword indices) in state I or J such that the codevectors[1] in these two states with the same binary codeword are close in the Euclidean sense. In an effort to do so, we define the distortion D' given by

$$D' = \sum_{\hat{s}} \sum_{s} \sum_{u} P_{s,\hat{s}} Pr(u|s) d(\beta(u,s), \beta(u,\hat{s})). \tag{A.5}$$

Note that D' measures the average Euclidean distance between codevectors with the same binary codeword of every possible pair of states that can be confused with each other.

Next we describe a heuristic algorithm for reindexing codevectors. In the following algorithm $\forall j \in \mathcal{S}$, $F[j] = 0$ means that codeword reassignment is not yet done in state j while $F[j] = 1$ means that codeword reassignment has been completed in state j.

Algorithm:

1. Compute $P_{i,j}$, $\forall i,j \in \mathcal{S}$. Set $m = 0$ (iteration index); $F[i] = 0$, $\forall i \in \mathcal{S}$. Compute D' using (A.5) and set $D^{(0)} = D'$, $P_{i,j}^{(0)} = P_{i,j}$, $\beta^{(0)} = \beta$.
2. Identify the pair $(I,J), I \neq J$, such that $P_{I,J} \geq P_{i,j}$, $\forall i,j \in \mathcal{S}$.

[1] In the algorithm that is described next, the codevectors are rearranged in one state (one of I or J) at a time; the state in which reassignment is done is denoted by s_c, while the one that is kept unchanged at that step is called s_h.

3. *If* $(F[I] = 0$ and $F[J] = 0)$

   then $s_h = I$ and $s_c = J$;

   else if $(F[I] = 1$ and $F[J] = 0)$

   then $s_h = I$ and $s_c = J$;

   else if $(F[I] = 0$ and $F[J] = 1)$

   then $s_h = J$ and $s_c = I$;

   else

   $m = m + 1$; $\beta^{(m)} = \beta^{(m-1)}$, $P_{i,j}^{(m)} = P_{i,j}^{(m-1)}$, $D^{(m)} = D^{(m-1)}$, $Pr^{(m)}(u|s) = Pr^{(m-1)}(u|s)$; go to step (6).

4. Determine $u^* \in \mathcal{N}$ such that

$$\sum_u Pr^{(m)}(u|I)d(\beta^{(m)}(u, s_h), \beta^{(m)}(u \oplus u^*, s_c)) = \min_{u_1 \in \mathcal{N}} \sum_u Pr^{(m)}(u|I)d(\beta^{(m)}(u, s_h), \beta^{(m)}(u \oplus u_1, s_c)), \quad \text{(A.6)}$$

   where $\oplus$ denotes the binary exclusive-OR (exor) operation.

   Set:

$$\beta(u, s_c) = \beta^{(m)}(u \oplus u^*, s_c), \ \forall u,$$

$$\beta(u, i) = \beta^{(m)}(u, i), \forall i \neq s_c, \forall u,$$

$$Pr(u|s_c) = Pr^{(m)}(u \oplus u^*|s_c), \ \forall u,$$

$$Pr(u|i) = Pr^{(m)}(u|i), \forall i \neq s_c, \ \forall u.$$

5. Compute D' (and hence $P_{i,j}$) using the β and $Pr(u|s)$ obtained in step (4);

   if $(D' < D^{(m)})$

   then $D^{(m+1)} = D'$, $m = m + 1$, $\beta^{(m)} = \beta$, $Pr^{(m)}(u|s) = Pr(u|s)$ and $P_{i,j}^{(m)} = P_{i,j}$;

   else

   $m = m + 1$, $D^{(m)} = D^{(m-1)}$, $\beta^{(m)} = \beta^{(m-1)}$, $Pr^{(m)}(u|s) = Pr^{(m-1)}(u|s)$ and $P_{i,j}^{(m)} = P_{i,j}^{(m-1)}$.

6. If $m < K$, go to step (2), else stop.

In step (4) of the algorithm, all the codevectors in a state codebook are changed by exoring with the same index[2] to ensure that the relative binary assignment (done using CO-VQ design algorithm of [27]) within each state codebook is not changed. The above algorithm increases robustness in the case when at any time the decoder state differs from the encoder state but the codeword is received correctly.

[2] u^* is estimated such that after the reindexing of the codevectors in s_c, the average Euclidean distance between the codevectors of I and J with the same codeword is minimized with the constraint that the relative binary assignment of the codevectors within each state is not changed.

Bibliography

[1] D.A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," *Proceedings of the IRE*, vol. 40, pp. 1098-1101, Sep. 1952.

[2] I.H. Wittin, R.M. Neal and J.G. Cleary, "Arithmetic Coding for Data Compression," *Communications of the ACM*, vol. 30, pp. 520-540, 1987.

[3] J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 530-536, Sep. 1978.

[4] Y. Linde, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84-95, Jan. 1980.

[5] W.H. Equitz, "A New Vector Quantization Clustering Algorithm," *IEEE Trans. on Acous., Speech and Signal Processing*, vol. 37, pp. 1568-1575, Oct. 1989.

[6] D.J. Vaishey and A. Gersho, "Simulated Annealing and Codebook Design," *Proc. IEEE ICASSP*, pp. 1176-1179, Apr. 1988.

[7] C. E. Shannon, "Coding Theorems for a Discrete Source with Fidelity Criterion," *IRE National Convention Record*, Part 4, pp. 142-163, Mar. 1959.

[8] R. M. Gray and Y. Linde, "Vector Quantizer and Predictive Quantizers for Gauss-Markov Sources," *IEEE Trans. Commun.*, vol. 33, pp. 855-865, Nov. 1987.

[9] B.H. Juang and A.H. Gray, "Multiple Stage Vector Quantization for Speech Coding," *Proc. IEEE ICASSP*, pp. 597-600, Apr. 1982.

[10] A. Gersho, "Asymptotically Optimal Block Quantization," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 373-380, Jul. 1979.

[11] J.H. Conway and N.J.A. Sloane, "Fast Quantizing and Decoding Algorithms for Lattice Quantizers and Codes," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 227-232, Mar. 1982.

[12] L.C. Stewart, R.M. Gray and Y. Linde, "The Design of Trellis Waveform Coders," *IEEE Trans. on Commun.*, COM-30, pp. 702-710, Apr. 1982.

[13] C. Bei, and R. M. Gray, "Simulation of Vector Trellis Encoding Systems," *IEEE Trans. Commun.*, vol. COM-34, pp. 214-218, Mar. 1986.

[14] B.H. Juang, "Design and Performance of Trellis Vector Quantizers for Speech Signals," *IEEE Trans. on Acous., Speech and Signal Processing*, vol. 36, pp. 1423-1431, Sep. 1988.

[15] V. Cuperman and A. Gersho, "Adaptive Differential Vector Coding of Speech," *Globe-Com-82*, pp. 1092-1096, Dec. 1982.

[16] P. Chang and R.M. Gray, "Gradient Algorithm for Designing Predictive Vector Quantizers," *IEEE Trans. on Acous., Speech and Signal Processing*, vol. 37, pp. 679-690, Aug. 1986.

[17] R. M. Gray, "Vector Quantization," *IEEE ASSP Mag.*, pp. 4-29, Apr. 1984.

[18] J. Foster, R.M. Gray and M.O. Dunham, "Finite-State Vector Quantization for Waveform Coding," *IEEE Trans. Inform. Theory*, vol. IT-31, pp. 348-359, May 1985.

[19] M. O. Dunham and R. M. Gray, "An Algorithm for the Design of Labeled-Transition Finite-State Vector Quantizers," *IEEE Trans. Commun.*, vol. COM-33, pp. 83-89, Jan. 1985.

[20] R. Aravind and A. Gersho, "Low-Rate Image Coding with Finite-State Vector Quantization," *Proc. IEEE ICASSP*, pp. 137-140, Apr. 1986.

[21] T. Kim, "New Finite State Vector Quantizers for Images," *Proc. IEEE ICASSP*, pp. 1180-1183, Apr. 1988.

[22] R. L. Baker and H. H. Shen, "A Finite-State Vector Quantizer for Low Rate Image Sequence Coding," *Proc. IEEE ICASSP*, pp. 760-763, May 1987.

[23] H. H. Shen, and R. L. Baker, "A Finite State/Frame Difference Interpolative Vector Quantizer for Low Rate Image Sequence Coding," *Proc. IEEE ICASSP*, pp. 1188-1191, Apr. 1988.

[24] P. A. Chou, T. Lookabaugh and R. M. Gray, "Optimal Pruning with Applications to Tree-Structured Source Coding and Modeling," *IEEE Trans. Inform. Theory*, vol. IT-35, pp. 299-315, Mar. 1989.

[25] A. Kurtenbach and P. Wintz, "Quantizing for Noisy Channels," *IEEE Trans. Commun. Technology*, vol. 17, pp. 291-302, Apr. 1969.

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

[27] N. Farvardin and V. Vaishampayan, "On the Performance and Complexity of Channel-Optimized Vector Quantizers," *IEEE Trans. Inform. Theory*, vol. IT-37, pp. 155-160, Jan. 1991.

[28] N. Phamdo, N. Farvardin and T. Moriya, "A Unified Approach to Tree-Structured and Multi-Stage Vector Quantization for Noisy Channels," submitted to *IEEE Trans. Info. Theory* in Dec. 1991.

[29] J.G. Dunham and R. M. Gray, "Joint Source and Noisy Channel Trellis Encoding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 516-519, Jul. 1981.

[30] E. Ayanoglu and R. M. Gray, "The Design of Joint Source and Channel Trellis Waveform Coders," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 855-865, Nov. 1987.

[31] N. Sugamura and N. Farvardin, "Quantizer Design in LSP Speech Analysis-Synthesis," *IEEE Journ. Selected Areas Commun.,* vol. 6, pp. 432-440, Feb. 1988.

[32] F. K. Soong and B. H. Juang, "Optimal Quantization of LSP Parameters Using Delayed Decisions," *Proc. IEEE ICASSP*, pp. 185-188, Albuquerque, NM, Apr. 1990.

[33] K. K. Paliwal and B. S. Atal, "Efficient Vector Quantization of LPC Parameters at 24 Bits/Frame," *Proc. IEEE ICASSP*, pp. 661-664, Toronto, Canada, May 1991.

[34] N. Sugamura and F. Itakura, "Speech Data Compression by LSP Speech Analysis-Synthesis Technique," *IECE Trans.*, vol. J64-A, pp. 599-605, Aug. 1981 (in Japanese).

[35] M. S. Schroeder and B. S. Atal, "Code-Excited Linear Prediction (CELP): High Quality Speech at Very Low Bit Rates," *Proc. IEEE ICASSP*, pp. 937-940, Tampa, FL, Mar. 1985.

[36] G. Davidson and A. Gersho, "Complexity Reduction Methods for Vector Excitation Coding," *Proc. IEEE ICASSP*, pp. 3055-3058, Tokyo, Japan, Apr. 1986.

[37] T. Moriya and M. Honda, "Transform Coding of Speech Using a Weighted Vector Quantizer," *IEEE Journ. Selected Areas Commun.*, vol. 6, pp. 425-431, Feb. 1988.

[38] J.P. Campbell, V.C. Welch and T.E. Tremain, "An Expandable Error-Protected 4800 BPS CELP Coder (U.S. Federal Standard 4800 BPS Voice Coder)," *Proc. IEEE ICASSP*, pp. 735-738, Glasgow, Scotland, May 1989.

[39] L.R. Rabiner and R. W. Scahfer, *Digital Processing of Speech Signals,* Prentice Hall, Englewood Cliffs, N.J., 1978.

[40] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*, the Wadsworth Statistics/Probability Series, Belmont, California: Wadsworth, 1984.

[41] R. Laroia and N. Farvardin, "A Structured Fixed-Rate Vector Quantization Derived from Variable-Length Encoded Scalar Quantizers," *Proc. of Twenty-Fourth Annual Conference on Information Sciences and Systems*, pp. 796-801, Princeton, NJ, Mar. 1990.

[42] R. Laroia, *Structured Fixed-Rate Vector Quantizers Derived from Variable-Length Encoded Scalar Quantizers*, Ph.D Dissertation, Electrical Engineering Department, Univ. of Maryland, College Park, MD, 1992.

[43] H. Abut, R. M. Gray and G. Rebolledo, "Vector Quantization of Speech and Speech-like Waveforms," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 423-435, Jun. 1982.

[44] A. Buzo, A. H. Gray, R. M. Gray and J. D. Markel, "Speech Coding Based upon Vector Quantization," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 562-574, Oct. 1980.

[45] N. M. Nasrabadi and R. A. King, "Image Coding Using Vector Quantization: A Review," *IEEE Trans. Commun.*, vol. 36, pp. 957-971, Aug. 1988.

[46] Y. Hussain and N. Farvardin, "Variable-Rate Finite-State Vector Quantization," *Proc. of Twenty-Fourth Annual Conference on Information Sciences and Systems*, Princeton, NJ, pp. 790-795, Mar. 1990.

[47] E. A. Riskin, E. M. Daly and R. M. Gray, "Pruned Tree-Structured Vector Quantization in Image Coding," *Proc. IEEE ICASSP*, Glasgow, Scotland, pp. 1735-1738, May 1989.

[48] R. L. Baker and R. M. Gray, "Differential Vector Quantization of Achromatic Imagery," *Proceedings of International Picture Coding Symposium*, Mar. 1983.

[49] Y. Hussain and N. Farvardin, "Adaptive Block Transform Coding of Speech Based on LPC Vector Quantization," *IEEE Trans. Acoust. Speech Signal Process*, vol. 39, pp. 2611-2620, Dec. 1991.

[50] A. N. Netravali and B. G. Haskell, *Digital Pictures Representation and Compression*, New York and London: Plenum Press, 1988.

[51] E. A. Riskin and R. M. Gray, "A Greedy Tree Growing Algorithm for the Design of Variable Rate Vector Quantizers ," *IEEE Trans. Signal Processing*, vol. 39, pp. 2500-2507, Nov. 1991.

[52] M.J. Mclaughlin and P.D. Rasky, "Speech and Channel Coding for Digital Land-Mobile Radio," *IEEE Journ. Selected Areas Commun.*, vol. 6, pp. 332-344, Feb. 1988.

[53] G.D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268-278, Mar. 1973.

[54] N. Phamdo, N. Farvardin and T. Moriya, "Combined Source-Channel Coding of LSP Parameters Using Multi-Stage Vector Quantization," *Research Meeting of the Institute of Electronics, Information and Communication Engineers* (IEICE, Japan), vol. SP90-52, DSP90-83, pp. 63-70, Nagoya, Japan, Oct. 1990.

[55] J. Hagenauer, "Rate-Compatible Punctured Convolutional Codes (RCPC Codes) and their Applications," *IEEE Trans. Commun.*, vol. 36, pp. 389-400, Apr. 1988.

[56] N. Tanabe and N. Farvardin, "Subband Image Coding Using Entropy-Coded Quantization over Noisy Channels," *Proc. IEEE ICASSP*, pp. 2105-2108, 1990.

[57] M Wang and T. Fischer, "Trellis Coded Quantization designed for Noisy Channels," submitted to IEEE Trans. Inform. Theory in May 1991.