Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

SIMPLE FAST VECTOR QUANTIZATION OF THE LINE SPECTRAL FREQUENCIES

by
Jin Zhou

Speech coding has been of interest to communication specialists for years. A number of technologies for describing and transmitting the speech spectral envelope have been studied. Among them, the Vector Quantization (VQ) of Line Spectral Frequencies (LSF) is receiving more attention because of its good rate-distortion performance. Typically, about 30 bits are assigned to code 10-dimensional LSF vectors with a resulting spectral distortion of less than 1dB. However, 30-bit full-search VQ is totally impractical in terms of both computational complexity and memory space. Various standard sub-optimal, low-complexity VQ techniques have been used for coding the LSF's in the literature. The most commonly used method is the split VQ, where the 10-dimensional LSF vector is typically partitioned into three sub-vectors of sizes 3, 3 and 4. Each sub-vector is then independently coded. This method reduces the complexity and required memory space significantly. However, the price paid is a compromise in performance. Reduced performance is inherent in most low-complexity VQ systems.

In this thesis we propose a simple fast-search VQ of the LSF's to be used on top of the split VQ (i.e., in each of the sub-vector domains). The main trait of the proposed method is that no sub-optimal codebooks are used and there is no further reduction in performance. In each sub-vector domain, a

full-size optimally trained codebook, typically of size 1024, is searched using a fast-search algorithm. The result of this search is identical to that of a full search, yet, only about 25% of full-search complexity is needed.

SIMPLE FAST VECTOR QUANTIZATION OF THE LINE SPECTRAL FREQUENCIES

by
Jin Zhou

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree
Master of Science in Electrical Engineering

Department of Electrical and Computer Engineering

May 1996

APPROVAL PAGE

SIMPLE FAST VECTOR QUANTIZATION OF THE LINE SPECTRAL FREQUENCIES

Jin Zhou

Dr. Ali N. Akansu, Thesis Advisor — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Yair Shoham, Co-Advisor — Date
Member of Technical Staff, Bell Laboratory, Lucent Technologies
Murray Hill, NJ

Dr. Nirwan Ansari, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Jin Zhou

Degree: Master of Science

Date: May 1996

Undergraduate and Graduate Education:

- Master of Science in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1996

- Bachelor of Science in Electrical Engineering,
  Shanghai University of Technology, Shanghai, P. R. China, 1990

Major: Electrical Engineering

Publications:

Jin Zhou, Yair Shoham and Ali Akansu,
"Simple and fast vector quantization of the linear spectral frequencies,"
The Fourth International Conference on Spoken Language Processing (ICSLP'96), Philadelphia, PA, October, 1996.

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Ali Akansu, who not only served as my advisor, but also constantly gave me support, encouragement, and reassurance. Special thanks is given to Dr. Yair Shoham for providing valuable and countless resources, insight, and intuition when I worked at the Lucent Technologies last summer. My sincere appreciation is also extended to Dr. Nirwan Ansari for his advice and interest as a committee member.

Many of my fellow graduate students in the Center for Communications and Signal Processing Research are deserving of recognition for their support. I also wish to thank Lisa Fitton for her proof-reading of this manuscript.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Voice Generation and Mathematical Model

Voice is one of the most important factors in communication between human beings. A sequence of voices composes speech signals. As the fundamental of the speech process, it is essential to understand the mechanism of speech production.

It is helpful to abstract the important features of the physical system in a manner that leads to a realistic yet tractable mathematical model. Figure 1.1 shows such a schematic diagram of the human vocal system. For completeness, the diagram includes the sub-glottal system, composed of the lungs, bronchi and trachea. This sub-glottal system serves as a source of energy for the production of speech. Speech is simply the acoustic wave that is radiated from this system when air is expelled from the lungs and the resulting flow of air is perturbed by constriction somewhere in the vocal tract.

Speech sounds can be classified into three classes: voiced, fricative (or unvoiced), and plosive sounds, according to their mode of excitation.

The vocal tract and nasal tract are shown in Figure 1.1 as tubes of a nonuniform cross-sectional area. As sound generated by sub-glottal system propagates down these tubes, the frequency spectrum is shaped by the frequency selectivity of the tube. Different sounds are formed by varying the shape of the vocal tract. Thus, the spectral properties of the speech signal vary with time as the vocal tract shape varies.

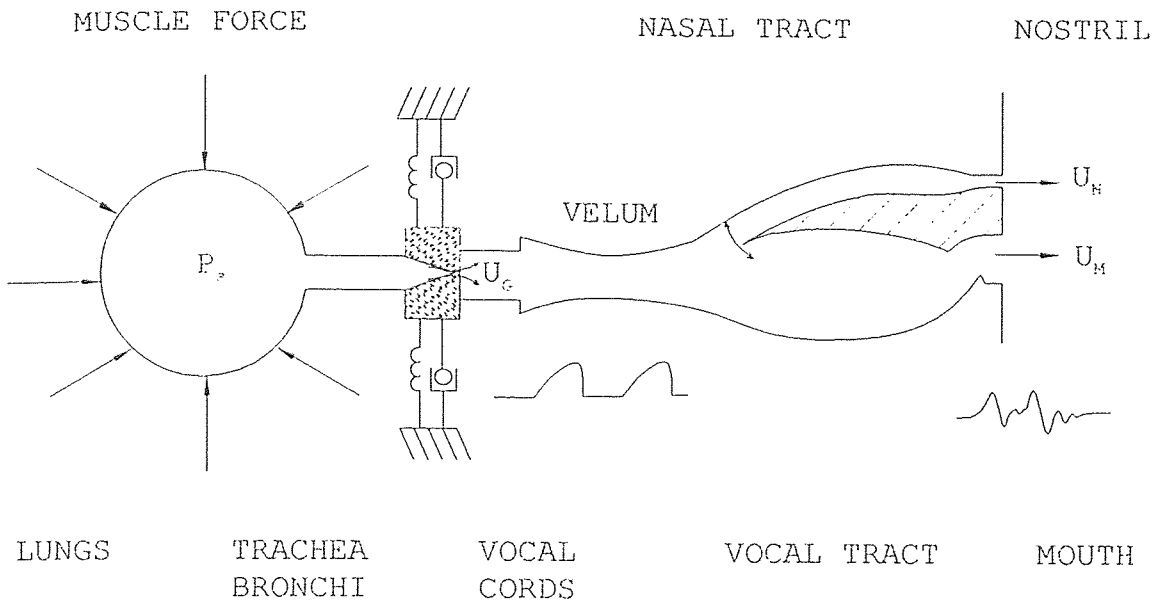

Figure 1.1 Schematized Diagram of the Vocal Apparatus

Based on the features of the acoustic theory of speech production, there are many detailed models for sound generation, propagation, and radiation. These models can in principle be solved with suitable values of excitation and vocal tract parameters to compute an output speech waveform. Indeed, it can be argued effectively that this may be the best approach to the synthesis of natural sounding synthetic speech. However, for many purposes such detail is impractical or unnecessary. In such cases, the acoustic theory points the way to a simplified approach of modeling speech signals. Figure 1.2 shows a general block diagram, representative of numerous models that have been used as the basis for speech processing. These models all have in common that excitation features are separated from the vocal tract and radiation features. The vocal tract and radiation effects are accounted for by the time-varying linear system. Its purpose is to model the resonance effects. The excitation generator creates a signal that is either a train of (glottal) pulses,

or randomly varying (noise). The parameters of the source and system are chosen so that the resulting output has desired speech-like properties.

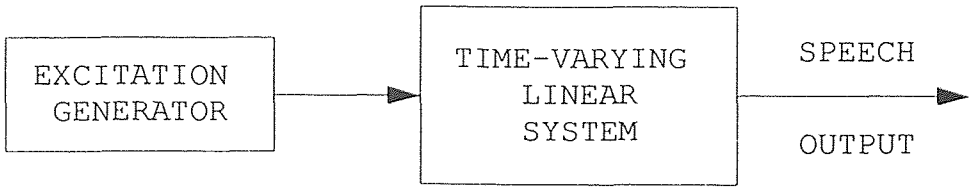

Figure 1.2 Source-System Model of Speech Production

1.2 Speech Coding

One of the earliest and most important applications of speech processing was *speech coding*, invented by Homer Dudley in the 1930's. After sampling the analog speech signal into s digital signal, a digital speech coder represents a discrete-time speech signal by a sequence of numbers in binary format. Thus, a finite number of bits (binary information units) per second is transmitted, which represents the speech signal. The receiver regenerates a replica of the speech signal from the bit sequence.

The purpose of speech coding is to reduce the bandwidth required to transmit the speech signal. In spite of the increased bandwidth provided by satellite, microwave, and optical communications system, the need to conserve bandwidth remains in many situations. The possibility of extremely sophisticated encryption of the speech signal is sufficient motivation for the use of digital transmission in many applications. Today, as wireless telephone has entered our society, both in social and business life, a need has arisen for these systems that digitize speech at as low a bit rate as possible, consistent with low terminal cost.

In the process of coding the speech signal to a finite resolution, distortion will be introduced. The objective in speech coding is to code the speech signal with as few bits per second as possible while maintaining an acceptable level of perceived distortion. Major breakthroughs in speech coding research in the 80s, notably the invention of the *code excited linear prediction* (CELP) coder, are the basis for current digital wireless telephony.

The prevailing type of speech coder today is based on a source-filter model, introduced in Chapter 1.1. In other words, speech production is modeled by an excitation signal fed through a digital filter. The excitation signal and the filter are coded separately. In this thesis, we are only concerned with the filter coding, which is spectral envelop coding. As a crucial factor for the perceived speech quality, this coding must be carefully performed.

1.2.1 Linear Prediction Coding (LPC)

The filter models the spectral envelope of the speech signal and is usually given as an all-pole filter obtained from linear prediction analysis. The optimal mth-order linear predictor is represented by an inverse filter:

$$A_{0m}(z) = 1 + a_{01}z^{-1} + a_{02}z^{-2} + \ldots + a_{0m}z^{-m}. \tag{1-1}$$

The order m of the filter is 10 in most CELP applications. To encode, the parameters $\{a_{01}, a_{01}, \ldots, a_{0m}\}$ are quantized to $\{a_1, a_2, \ldots, a_m\}$, introducing a filter

$$A_m(z) = 1 + a_1z^{-1} + a_2z^{-2} + \ldots + a_mz^{-m}.$$

Such a speech coding technique is called *linear predictive coding* (LPC), and the $\{a_i\}$s are *LPC parameters*. Due to the quasi-stationary

character of the speech signal, the LPC filter is updated regularly on a frame basis, with a typical frame length of 20 ms. Thus, for each speech frame, a description of the LPC filter must be communicated to the receiver. The process of coding the LPC filter at a finite number of bits per frame is known as *LPC coding*, *LPC spectrum coding*, or *LPC quantization*.

Although the linear prediction analysis gives a simple filter model, in speech coding the LPC parameters are known to be inappropriate for quantization because of two important reasons. One is that the LPC parameters are too dynamic to be used efficiently in speech coding. The other is that the roots of the LPC analysis filter are located inside the unit circle of the z-plane. After speech coding, the roots may be out of the unit circle, which results in an instability problem.

1.2.2 Line Spectral Frequency (LSF)

Different sets of parameters representing the same spectral information, such as *reflection coefficients* and *log area ratios*, etc., were thus proposed for quantization in order to alleviate the above-mentioned problems. LSP is one such representation. It was first introduced by Itakura in 1975 [1] as *line spectral representation* (LSR), and later developed into the *line spectrum pair* (LSP). In 1984, Soong and Juang of Bell Laboratories published a more detailed paper concerning LSP properties [2].

For a given LPC polynomial as Equation (1-1), we can construct two artificial $(m+1)$th-order polynomials by setting the $(m+1)$-th reflection coefficient, k_{m+1}, to be +1 or -1. These two cases correspond, respectively, to an entirely closed or an entirely open end at the last section of an acoustic

tube of m+1 piecewise-uniform sections. Let $P(z)$ be the symmetric polynomial (k_{m+1} = 1) and $Q(z)$ be the antisymmetric polynomial (k_{m+1} = -1):

$$\left.\begin{matrix} P(z) \\ Q(z) \end{matrix}\right\} = A_{0m}(z) \pm z^{-(m+1)} A_{0m}\left(z^{-1}\right).$$

It is obvious that

$$A_{0m}(z) = \frac{1}{2}\left[P(z) + Q(z)\right].$$

It has been proved that the LSP polynomials, $P(z)$ and $Q(z)$, have the following interesting properties: (1) all zeros of LSP polynomials are on the unit circle, (2) zeros of $P(z)$ and $Q(z)$ are interlaced, (3) the minimum phase property of $A_{0m}(z)$ can be easily preserved if the first two properties are intact after quantization -- all zeros of $A_m(z)$ are inside the unit circle.

Line spectral frequency (LSF) parameters, $\{\omega_i : i = 1,2,\ldots,m\}$, are the roots of $P(z)$ and $Q(z)$ in $0 < \omega < \pi$. Odd LSFs are the roots of $P\left(e^{-j\omega}\right)$ and even LSFs are the roots of $Q\left(e^{-j\omega}\right)$. the LSFs have the fundamental property $\omega_i > \omega_{i-1}$, i=2,...,m, to guarantee the stability of the LPC analysis filter. The smaller dynamic range of LSFs $(0 - \pi)$ is also a good start for quantization.

Generally, LSF parameters have both a well-behaved dynamic range and a filter stability preservation property, and can be used to encode LPC spectral information even more efficiently than many other parameters.

CHAPTER 2

QUANTIZATION OF LSF

Various quantization techniques have been developed for LSFs during the last decade, including many *scalar quantization* (SQ) and *vector quantization* (VQ) methods.

Quantization is the heart of analog-to-digital conversion. In its simplest form, a quantizer observes a single number and selects the nearest approximating value from a predetermined finite set of allowed numerical values.

2.1 Scalar Quantization

Scalar quantization is a one-dimensional quantization. An N-point scalar quantizer Q can be defined as a mapping Q: $R \rightarrow C$, where R is the real line and

$$C \equiv \{y_1, y_2, y_3, \ldots, y_N\} \subset R$$

is the *output set* or *codebook* with size $|C| = N$. The *output values*, y_i, are sometimes referred to as the *output level*. For an input, the corresponding output is the value rounded to the nearest y_i.

Every quantizer can be viewed as the combined effect of two successive operations (mappings), an *encoder*, E, and a *decoder*, D. The encoder is a mapping E: $R \rightarrow I$, where $I = \{1,2,3,\ldots N\}$, and the decoder is the mapping D: $I \rightarrow C$. Thus if $Q(x) = y_i$, then $E(x) = i$ and $D(i) = y_i$. With the definitions, we have $Q(x) = D(E(x))$. In the context of a waveform communication

system, the encoder transmits the index I of the selected level, y_i, chosen to represent an input sample, and not the value y_i itself. The decoder can be implemented by a table-lookup procedure, where the table or codebook contains the output set, which can be stored with extremely high precision. A sequence of reproduction values obtained in this manner provides an approximation to the original sequence of samples and hence, using a suitable interpolating filter, an approximation to the original waveform can be reconstructed.

2.1.1 The Uniform Quantizer

The most common of all scalar quantizers is the *uniform quantizer*, sometimes called a "linear" quantizer because its staircase input-output response lies along a straight line (with unit slope).

A uniform quantizer is a regular quantizer in which the boundary points are equally spaced and the output levels for cells are the midpoints of the quantization interval.

2.1.2 The Nonuniform Quantizer

A general model for a nonuniform quantizer with a finite number of levels is shown in Figure 2.1. The input x is first transformed with a memoryless monotonic nonlinearity G (compressor) to produce output $y = G(x)$, then it is quantized with a uniform quantizer producing $\hat{y}$, and finally it is transformed with the inverse nonlinearity G^{-1} (expandor). The final (nonlinearly) quantized output is $\hat{x} = G^{-1}(\hat{y})$.

For a nonuniform quantizer, the step size, $\Delta_i \equiv x_i - x_{i-1}$, varies from one cell to another of the partition.

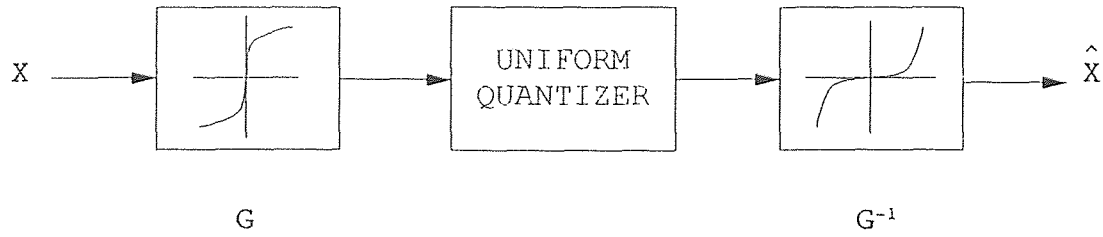

Figure 2.1 Compandor Model of Uniform Quantization

There are two major advantages to using nonuniform spacing of quantization levels. First, it is possible to significantly increase the dynamic range that can be accommodated for a given number of bits of resolution by using a suitably chosen nonuniform quantizer. Second, it is possible to design a quantizer tailored to the specific input statistics so that a considerably superior SNR is attained for a given resolution and given input pdf when the levels are allowed to be nonuniformly spaced.

2.1.3 SQ of LSF

An efficient SQ of LSF was found by Soong and Juang in 1984 [2]. Instead of the LSF itself, they noticed the nonuniform statistical distributions and spectral sensitivities of adjacent LSF differences. Based upon these two nonuniform properties, a novel, globally optimal scalar quantizer is designed for each differential LSF.

The distributions of the LSFs are plotted in Figure 2.2(a) in histogram form. The LSFs, f_i's, in the figures have been normalized. It is clear from

the figure that the distribution range varies from one LSF histogram to another. To reduce this variability and the associated spectral sensitivity Soong and Juang have proposed a differential coding scheme in which, instead of the absolute values of the LSFs, the differences between adjacent LSFs are encoded. The motivation for using a differential scheme is that the LSF differences are observed to be less divergent than the absolute frequencies themselves. Histograms of the LSF differences are plotted in Figure 2.2(b) to illustrate this point. A reduced dynamic range of the differential parameters is apparent by comparing Figure 2.2(a) with Figure 2.2(b).

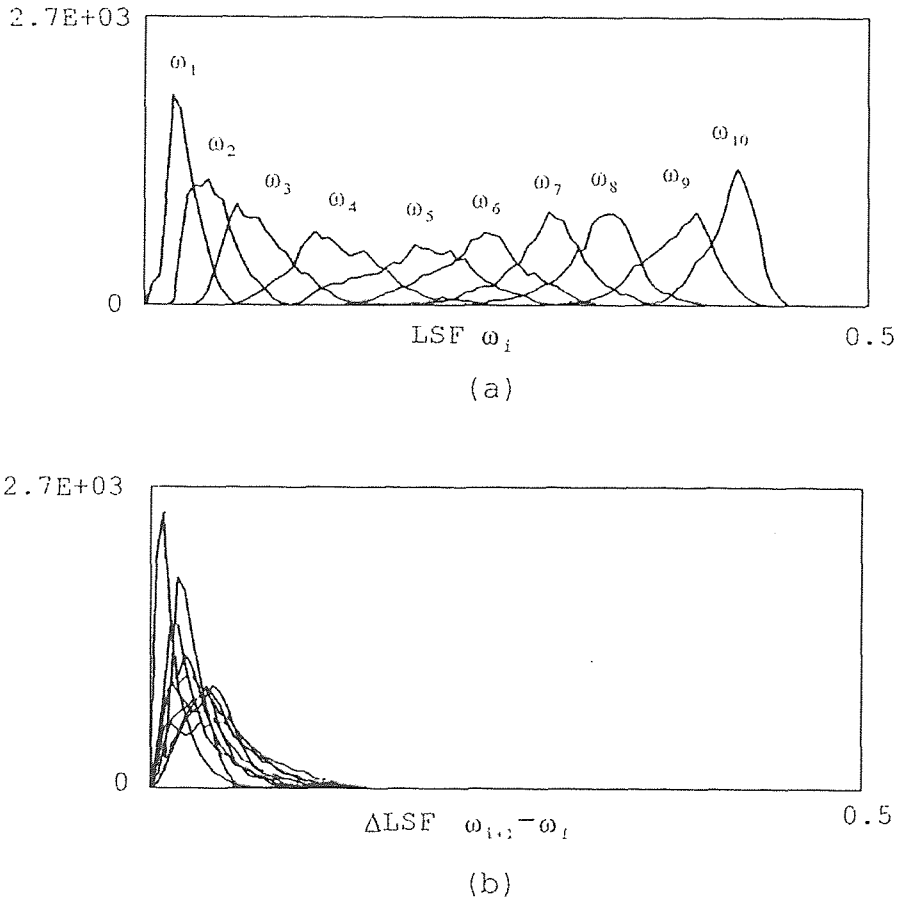

Figure 2.2 (a) Histograms of LSF
(b) Histograms of LSF Differences

The quantization performance is usually measured by the *log spectral distortion* S, which is defined as

$$S^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi}\left(20\log_{10}\left|\frac{A_0(\omega}{A_m(\omega)}\right|\right)^2 d\omega,$$

and a commonly accepted level for reproducing perceptually transparent spectral information is less than 1dB. The SQ method achieves a 1dB average log spectral distortion at 32 bits/frame.

2.2 Vector Quantization

Vector quantization (VQ) is a generalization of scalar quantization to the quantization of a vector, an ordered set of real numbers. While scalar quantization is used primarily for analog-to-digital conversion, VQ is used with sophisticated digital signal processing, where in most cases the input signal already has some form of digital representation and the desired output is a compressed version of the original signal.

A vector quantizer Q of dimension k and size N is a mapping from a vector in k-dimensional Euclidean space, R^k, into a finite set, C, containing N output or reproduction points called *code vectors* or *codewords*. Thus,

$$Q: \quad R^k \rightarrow C,$$

where codebook $C = \{y_1, y_2, ..., y_N\}$ and $y_i \in R^k$ for each $i \in J \equiv \{1,2,...,N\}$. Associated with every N point vector quantizer is a partition of R^k into N regions or *cells*, R_i for $i \in J$.

A vector quantizer can be decomposed into two component operations, the *vector encoder* and the *vector decoder*. The encoder E is the mapping from

R^k to the index set J, and the decoder D maps the index set J into the reproduction set C. Thus,

$$E:\ R^k \rightarrow J \text{ and } D:\ J \rightarrow R^k .$$

A given partition of the space into cells fully determines how the encoder will assign an index to a given input vector. On the other hand, a given codebook fully determines how the decoder will generate a decoded output vector from a given index. The overall operation of VQ can be regarded as the cascade or composition of two operations:

$$Q(x) = D \cdot E(x) = D(E(x)).$$

Figure 2.3 illustrates how the cascade of an encoder and decoder defines a quantizer.

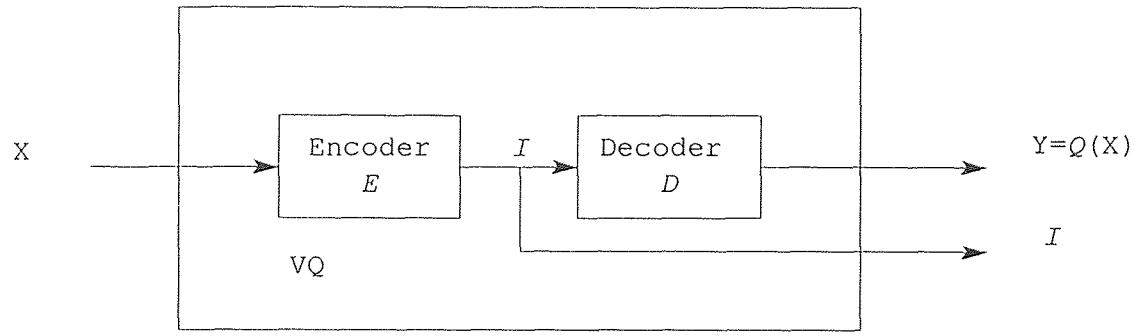

Figure 2.3 A Vector Quantizer as the Cascade of an Encoder and a Decoder

In a digital speech communication system, the encoder of a vector quantizer performs the task of selecting (implicitly or explicitly) an appropriately matching code vector y_i to approximate, or in some sense to describe or represent, an input vector x. The index I of the selected code vector is transmitted (as a binary word) to the receiver, where the decoder

performs a table-lookup procedure and generates the reproduction y_i, the quantized approximation of the original input vector.

Figure 2.4 shows an intuitional two-dimensional case of a vector quantizer operation geometrically. This quantizer, whose bounded cells are polygons, assigns any input point in the plane to one of a particular set of N points or locations in the plane.

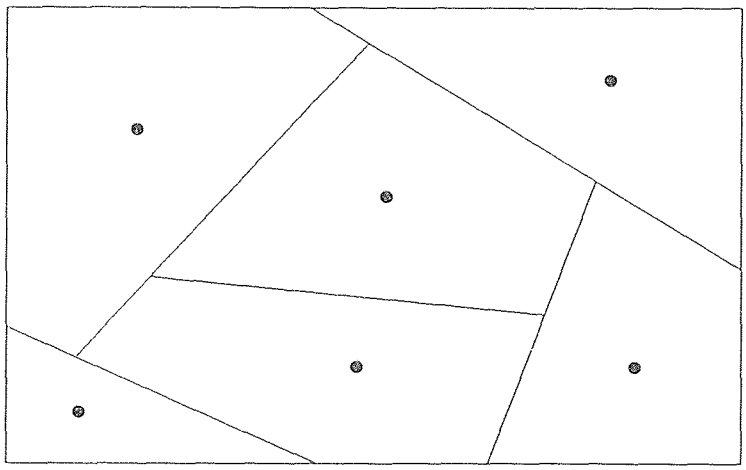

Figure 2.4 A Regular Vector Quantizer

In fact, the term "vector quantizer" is commonly assumed to be synonymous with "nearest neighbor vector quantizer," which, having the feature that the partition is completely determined by the codebook and a distortion measure.

Suppose that *d(x,y)* is a distortion measure on the input/output vector space, the ubiquitous squared error distortion measure defined by the squared Euclidean distance between the two vectors:

$$d(x, y) = \|x - y\|^2 = \sum_{i=1}^{k} (x_i - y_i)^2 .$$

In speech coding, it has already been proved that input-dependent weightings are useful and often only slightly more complicated. This makes the distortion measure become:

$$d(x,y) = \sum_{i=1}^{k} w_i (x_i - y_i)^2 .$$

A nearest neighbor vector quantizer is defined as one whose partition cells are given by

$$R_i = \{x : d(x,y_i) \le d(x,y_j) \text{ all } j \in J\} .$$

In other words, with a nearest neighbor encoder, each cell R_i consists of all points x that have less distortion when reproduced with code vector y_i than with any other code vector. The encoder process is illustrated in Figure 2.5, where $c(.,.)$ represents the functional operation.

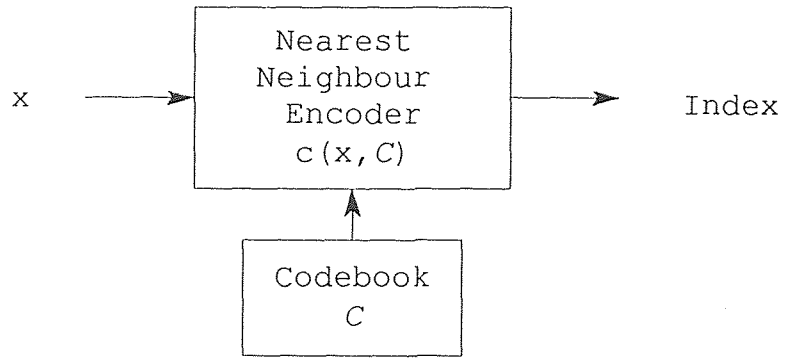

Figure 2.5 Nearest Neighbor Encoder with a Codebook ROM

There is no better way to quantize a single vector than to use VQ with a codebook that is optimal for the probability distribution describing the random vector. VQ considers the entire set of LSF parameters as an entity

and allows for direct minimization of quantization distortion. Because of this, the vector quantizers result in smaller quantization distortion than the scalar quantizers at any given bit rate. The demand for higher performance at lower bit rates has shifted the focus of coding and transmitting the LSF's to the use of more powerful, yet more complex, vector quantization (VQ) techniques. Typically, about 30 bits are assigned to code 10-dimensional LSF vectors with a resulting spectral distortion of less than 1dB.

However, the directly use of 30-bit full-search VQ is totally impractical in terms of both computational complexity and memory space. Such a vector quantizer has the following problems. First, a large codebook requires a prohibitively large amount of training data (a set of observations/samples of the signal to be quantized) and the training process can take too much computation time. Second, the storage and computational requirements for vector quantization encoding will be prohibitively high. Unconstrained VQ is severely limited to rather modest vector dimensions and codebook sizes for practical problems.

Various standard sub-optimal low-complexity vector quantizer has to be used for getting transparent quantization of LSF information. These techniques have been developed to apply various constraints to the structure of the VQ codebook and yield a correspondingly altered encoding algorithm and design technique. To reduce the cost of VQ, two general approaches have been used in the past. First, storage can be reduced by combining several smaller codebooks. Best known among these are the *multi-stage VQ* and the *split VQ* [3], which exemplify schemes where the overall quantization is divided into smaller tasks or the codebook is organized so that it can be

handled. Second, to reduce computational complexity, various structured codebook VQ methods have been developed, such as *classified VQ* [4]. These methods generally compromise the performance achievable with unconstrained VQ, but often provide a very useful and favorable trade-off between performance and complexity.

Design of VQ codebooks in an application such as quantization of speech parameters is usually accomplished by an iterative training algorithm. A training database of representative source vectors is compiled and the codebook is optimized for this database with a suitable distortion measure. The most widely used algorithm is the LBG algorithm, which is beyond the scope of this thesis.

In this chapter, the most popular constrained VQ method, namely split VQ, is introduced. This scheme is an example of the so-called *Produce Code VQ*, where the overall codebook is built as a Cartesian product of several smaller codebooks. A synthesis function generates the overall reconstruction vector from a set of vectors from the smaller codebooks. Then classified VQ is introduced.

2.2.1 Split VQ

Paliwal and Atal presented their split VQ scheme, obtaining an average spectral distance of 1dB at 24bits/frame in 1991 [3]. This work and the good results reported has spurred researchers within this area. Today this work is often used as a benchmark for comparing other results.

In split VQ, the LSF vector is split into a number of parts and each part is quantized separately using vector quantization.

As a scalar quantization is described by

$$Q(x) = \sum_{i=1}^{N} y_i S_i(x),$$

where $S_i(x) = \begin{cases} 1 & x \in R \\ 0 & otherwise \end{cases}$

is a selector function.

The split VQ can be described as

$$Q(X) = \begin{bmatrix} Q(X_1) \\ Q(X_2) \\ \cdots\cdots \\ Q(X_p) \end{bmatrix}, \quad p < k,$$

where the overall quantization problem is partitioned into p parts or splits, and each small quantizer is $Q(X_j) = \sum_{i=1}^{N_1} y_{i1} S_i(X_j)$. The synthesis function simply concatenates vectors from the individual codebooks. The split VQ enables us to design and handle the individual codebooks in parallel. We have the possibility of exploiting dependence between components considered by the same quantizer but not dependencies between components represented in different codebooks.

When first introduced, split VQ was used in LPC quantization. We know that the split vector quantizer reduces the complexity at the cost of degraded performance. Thus there is a trade-off in complexity and performance, which determines the number of parts to be made for split vector quantization. Usually $10th$-order LPC parameter vector is divided into two parts. Typically for a $10th$-dimensional LSF, the vector is

partitioned into three sub-vectors containing the first 3, second 3, and last 4 LSFs, respectively. Each sub-vector is then independently coded.

2.2.2 Classified VQ

Ramamurthi and Gersho introduced classified VQ in 1986 [4]. At that time, this technique was used in image processing to detect the edges out of the regular pixels.

Classified VQ (CVQ) is based on an heuristic characteristic that the designer adopts to identify the mode of the particular input vector. Thus instead of a test codebook, an arbitrary classifier may be used to select a particular subset of the codebook to be searched. Varying size subsets is allowed and we can partition the codebook into unequal-sized small codebooks, called *classes*. Figure 2.6 illustrates this scheme. The classifier generates an index, an integer from 1 to m, which identifies the class C_i to search for a nearest neighbor. The codeword consists of the index i. which specifies the m codebooks to be selected.

Many possibilities exist for the choice of a classifier. It can be a simple VQ encoder that identifies the m regions of the input space where the current input vector lies. In this case, CVQ is very similar to a two-stage VQ, where the second stage nodes may each have different codebook sizes. The classifier in a CVQ encoder can extract one or more features from the input vector, such as the mean (average amplitude of the samples), the energy (sum of the squares of the samples), the range (difference between maximum and minimum amplitudes of the components), and other statistics.

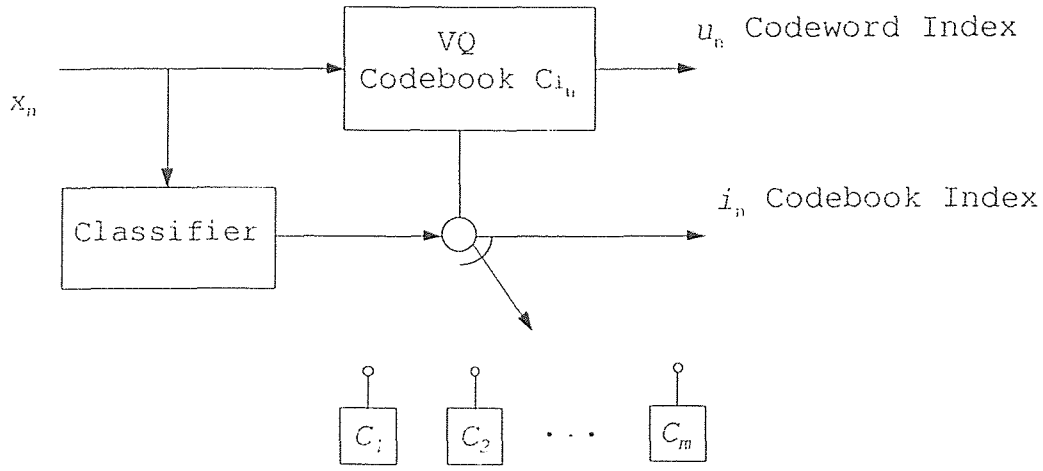

Figure 2.6 Classified VQ

The particular physical meaning of the vector may lead to many other features. As first introduced in image processing, in a two-dimensional set of pixels representing an image block, an edge detector, can be used to identify the presence or absence of edges (or high detail) and the direction and location of edges if present. For a vector that represents a block of consecutive speech samples, the zero-crossing count (number of sign changes in proceeding from the first to the last sample) can be a useful feature. Once such features are identified, the task remains to decide how many distinct modes should be used. A scalar or vector classifier must be chosen or designed to provide a complete classification operation. The classifier can be designed either heuristically or using the LBG with a training set of features that are extracted from a training set of input vectors.

2.3 Problem to Deal With

In this study, we try to find an optimal low-complexity, low-memory vector quantization for a 10th-order LP model.

Split VQ is considered as a good development of low-complexity, low-memory spectrum quantization. However, unstructured full-search VQ of the LSF sub-vectors is still too complex for many applications.

The problem addressed in this thesis is the reduction of the LSF split VQ complexity without a compromise in performance. The fast VQ methods proposed here are applied to each sub-vector independently and achieve the same level of performance at about 25% of the full-search split VQ complexity. The proposed method is based on classified VQ (CVQ).

CHAPTER 3

CLASSIFIED VQ ON A SPLIT BASIS

Codebook designing is beyond the scope of this thesis. All the work is based on a codebook designed by Dr. Yair Shoham, of Lucent Technologies (formerly AT&T Bell Laboratories). Some necessary background is introduced here.

A training set of size 200,000 is used to generate a codebook, and the quantization performance is measured by a test file of size 20,000 in term of the spectral distortion.

The entire 10th-order LSF vector in each frame is divided into 3 sub-vectors which contain 3-, 3-, and 4-dimensions respectively. Each sub-vector is then independently coded using 10-bit. These codebooks are designed to minimize the weighted squared error between the training set and code vectors:

$$\min \sum_{i=i_1}^{i_2} w_i (x_i - y_i)^2 , \qquad 1 \le i_1 , \; i_2 \le 10 ,$$

where i_1 and i_2 are the head and end of each sub-vector, respectively, x_i's are the training set, y_i's are the code vector, and w_i's are weights.

Adding weights is a simple but useful modification of the squared error distortion that allows a different emphasis to be given to different vector components. Regularly, the weights are inversely proportional to the distances between adjacent vector components, so the code vector is more sensitive to closer pairs. Two codebooks using different weights are designed

and their quantization performances are compared. The first weight is defined as

$$w_i = \frac{1}{z_i} + 1, \text{ for } 1 \le i \le 10,$$

where $z_i = \min\{x_{i+1} - x_i, x_i - x_{i-1}\}$ for $i = 2,3,\cdots,9$

and

$$z_1 = x_2 - x_1,$$

$$z_{10} = x_{10} - x_9.$$

The codebook designed by this weight gives a full-search spectral distortion of 0.767dB on the test file.

The second weight is defined as

$$w_i = \frac{1}{(x_{i+1} - x_i)(x_i - x_{i-1})}, \text{ for } i = 1,\cdots,10,$$

where $x_0 = 0$, $x_{11} = \pi$ are not real LSF parameters. This weight gives the test file a lower spectral distortion of 0.757dB, and is taken for the later use.

Compared with coding large amount of LSFs, the design of the codebook is only a one-time job. Especially with the development of today's computer technology, the design complexity is not a primary issue in practice. Meanwhile as the requirement on speech communication increases, LPC filter coding, which needs to be updated each 20 ms, involves greater calculation. The anxious to process speech signals at higher speed becomes more and more imperative. This thesis attempts to reduce the search complexity by applying a simple fast classifier.

The new search scheme consists of two stages. The first-stage is a simple classification. The input vector if determined to belong to a certain

class out of a predetermined number of classes. Each class is represented by a small set of code vectors, which is a segment (C_i in Figure 2.6) in the optimal codebook. The union of all sets (classes) form the optimal codebook and the sets may overlap.

Second-stage is a small range full-search VQ in those classes. Once the class is determined, a full-search is conducted over the set of that class and possibly over a few neighboring sets. The set to be searched is small, yet, the optimal code vector (the one selected by a full-search) is included (and selected) with extremely high probability.

Note that CVQ requires about the same or even slightly more memory space since the fully optimal main codebook is used. The index output of the second-stage search is then sent in a binary form to the receiver side.

For easy access and search, each sub-vector codebook needs to be rearranged. After a classifier is chosen, the code vectors are classified. Those belonging to the same class are gathered, and given the same class index. The numbers of the code vectors in each class are saved, and referred to as an index table, so that in the later second-stage full-search one can tell the boundary between classes. Figure 3.1 illustrates the process to classify a codebook.

Usually the first-stage classification is a simple quantization, either a scalar quantization or a simple vector quantization.

It is obvious that the search complexity decreases when class number increases. But this does not mean the codebook may be partitioned into as many subsets as possible. In a small subset-size situation, usually more candidates are necessary for an optimal result. Meanwhile, small size

classes bring more boundary standards to storing and it increases storage cost. There is a trade-off between the class size and number of candidate.

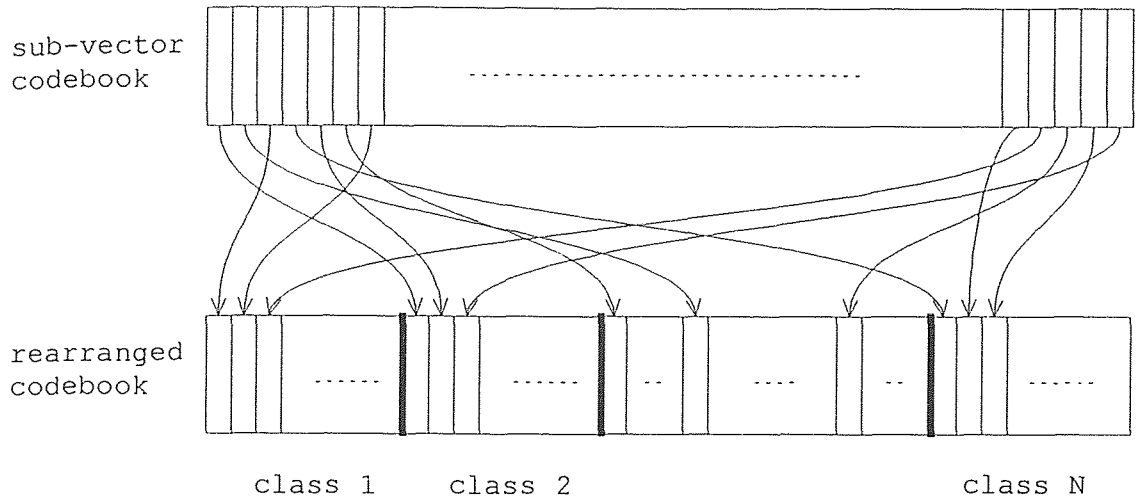

Figure 3.1 Classified VQ Based on Split Basis

An expected classifier should have these characteristics: (1) fast, (2) small search range, (3) limited class candidate.

The design of a VQ classifier includes 3 steps: (1) find a fast classifier, (2) partition the codebook into classes, (3) rearrange the codebook for easy access to classes.

Usually, as a cost of this search scheme, either the quantization performance is sub-optimal or more memory is involved.

The following chapters describe the proposed classifiers and CVQ systems. The systems are characterized by complexity, memory and performance. The complexity unit is the effort required to computer a weighted distance per dimension, which for VQ is $w \times (x-y)^2$, and for SQ is $\|x-y\|$. So the complexity of a k-dimensional VQ of size N is $k \times N$. The memory is given in terms of computer words. The performance is measured

by average spectral distortion in dB. Another performance figure is the number of miss-quantized vectors, namely, different from the ones obtained by full-search VQ. In this thesis, we assume a 3, 3, 4 split of the 10-dimensional LSF vector and a codebook of 1024 code vectors for each LSF sub-vector, so the full-search complexity is $1024 \times (3+3+4) = 10240$. Extensions can obviously be made for different dimensions and sizes.

In the following description we refer to codebooks on three levels. We have a codebook for the entire constrained VQ scheme, which we refer to as the *main codebook*. The smaller codebooks that are designed and handled in the structurally constrained VQ, namely all three sub-vector codebooks, are referred to as the *individual codebooks*. After classify all the smallest codebooks are called *classes*.

Although the various-sized classes are allowed, it is highly recommended to partition the codebook into equal-sized classes. For an unknown input vector, the possibility that it lies in different subsets of the input space is equal. In certain candidate situations, even distribution between classes gives each input vector an equal search range, which means not only equal opportunity to find a code vector, but also average complexity regardless of which class it belongs to. This brings a better overall quantization performance.

CHAPTER 4

VQ-BASED CLASSIFIER WITH CLASS CODEBOOK

4.1 Definition

In vector quantization, each code vector is a representative of its nearest vectors. The VQ-based classifier is an extension of this idea--it finds the representative of the nearest code vectors. These new representative, called *class vectors*, are designed by the same algorithm, from the same training set, as the individual codebook but they use fewer bits. For example, 4, 5, and 6 bits for each sub-vector have 16, 32, and 64 codes, respectively. These codes constitute a small VQ codebook--a *class codebook*. As mentioned in Chapter 2.2, if the full-search vector quantization Q_1 is a mapping from k-dimensional Euclidean space, R^k, into an N output individual codebook C_1 as

$$Q_1 : R^k \rightarrow C_1,$$

where $C_1 \equiv \{y_1, y_2, \cdots, y_N\}$ and $y_i \in R^k$ for each $i = \{1,2,\cdots,N\}$, then the first-stage vector quantization Q_2 is a mapping from R^k into class codebook C_2 as

$$Q_2 : R^k \rightarrow C_2,$$

where $C_2 \equiv \{y'_1, y'_2, \cdots, y'_M\}$ and $y'_j \in R^k$ for each $j = \{1,2,\cdots,M\}$, $M < N$. While the full-search vector quantizer assigns any input vector to one of a particular set of N outputs, the first-stage of VQ-based classifier assigns any input vector to one of a particular set of M outputs, larger than the previous

one. The class codebook is smaller than the individual codebook but it is also more representative.

After the classifier is chosen, all the code vectors y_is go through a VQ filter and compared to each class vector y_j'. Suppose that $d(y_i, y_j')$ is a distortion measure on the code vector/class vector space, defined as

$$d(y_i, y_j') = \|y_i - y_j'\|^2 = \sum_{n=1}^{k} (y_{in} - y_{jn}')^2 .$$

A nearest neighbor VQ finds the y' index I_i where

$$d(y_i, y_{I_i}') \le d(y_i, y_j')$$

all $j \in [1, M]$. The precise distance $d(y_i, y_j')$ should be defined as

$$d(y_i, y_j') = \sum_{n=1}^{k} w_i (y_{in} - y_{jn}')^2 .$$

As a roughly classifier, however, it does not matter whether the weights are used or not. To be simple, the weights are omitted in the first-stage quantization.

After classification, each code vector is given a particular class index I_i. Those code vectors having the same class index, say 3, are nearest to the same class vector y_3', and are placed side by side in the new codebook. The order of the class vectors decides the order of the class; namely, the first class having all the code vectors nearest to first class vector. With N class vectors the individual codebook is partitioned into N classes. While in full-search VQ code vectors are centroids of their nearest training vectors, in this first-stage VQ these class vectors are centroids of their nearest code vectors. As in

Figure 4.1, a two-dimensional case is illustrated. The dots represent code vectors in an individual codebook, the xs represent class vectors.

After classifying and rearranging the codebook, the number of code vectors in each class must be stored as an index table. Later, when we find the nearest m classes to an input, the table may lead to the starting and ending search positions in the individual codebook. Both the class codebook and the index table need to be saved. The new codebook is:

codebook = index table + class codebook + main codebook

and its structure is illustrated in Figure 4.2. For N classes case, the added memory (index table and class codebook) is $3 \times N + 10 \times N = 13 \times N$ words.

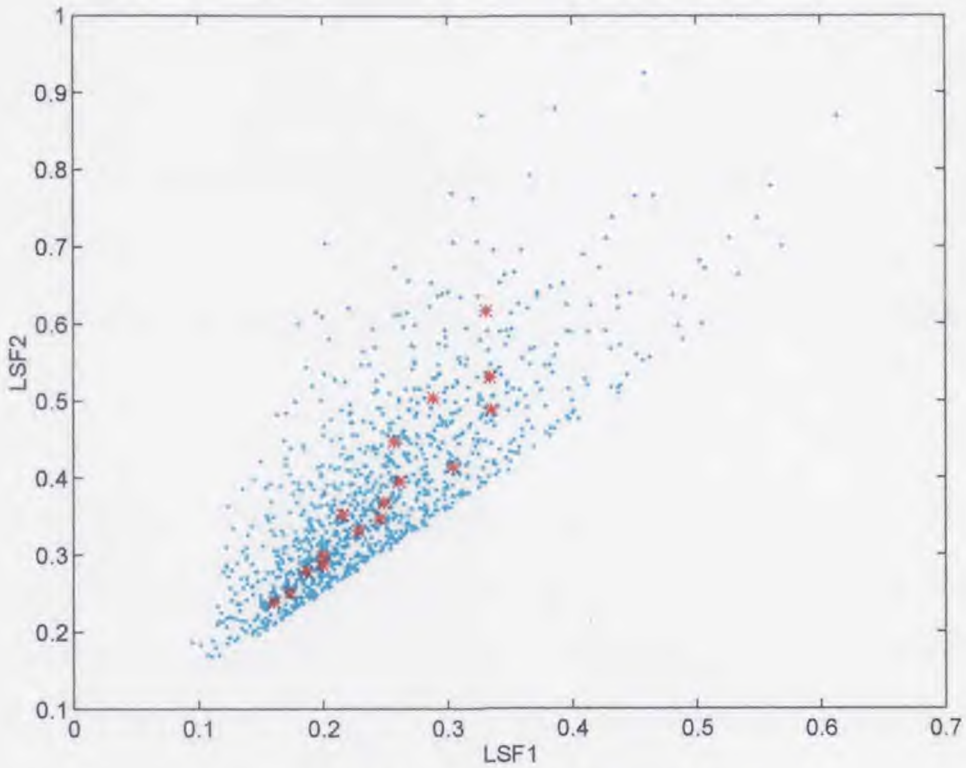

Figure 4.1 VQ-Based Classifier

In quantization, an input vector x goes through the first-stage classifier, compares to each class vector and finds the m nearest class indices with m minimum $d(x, y_j')$ defined as

$$d(x, y_j') = \|x - y_j'\|^2 = \sum_{n=1}^{k} (x_n - y_{jn}')^2 .$$

The index table may indicate the starting and ending positions of these classes. The next step is a full-search VQ among the code vectors in these classes.

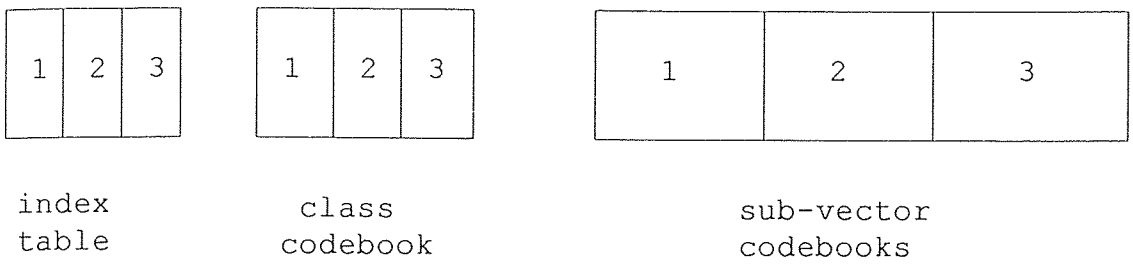

Figure 4.2 Codebook Structure of VQ-Based Classifier

The new search complexity for N class, m candidates is roughly

$$10 \times N + \frac{10 \times 1024}{N} \times m .$$

Obviously, if the individual codebook is partitioned into more classed, the complexity for a search in one candidate must be lower. But meanwhile, because of fewer code vectors to choose from, the quantization performance becomes worse. The simulation test attempts to find the best compromise between the small class size, fewer candidates, and better quantization performance.

Because the class vectors are designed by the same algorithm as the individual codebook, it is clear that each class should contain roughly an equal number of code vectors, which is a premise for a good classifier.

The disadvantage of this scheme is that for each different class size, new class codebooks need to be designed. Correspondingly, the codebook needs to be classified and rearranged. Although it is a one-time job, more flexibility is desired. The memory cost of this scheme includes an index table and class codebooks, both which increase when a smaller class size is taken.

4.2 Simulation Results

We partition each individual codebook into 16-, 32-, and 64-class. The number of candidates are taken from 1 and added until the spectral distortion is as good as the full-search result 0.757dB. The new quantization result is compared to the full-search quantization result. Figure 4.3 is the histogram of code vectors between classes ($n = 16$), which shows an almost even distribution.

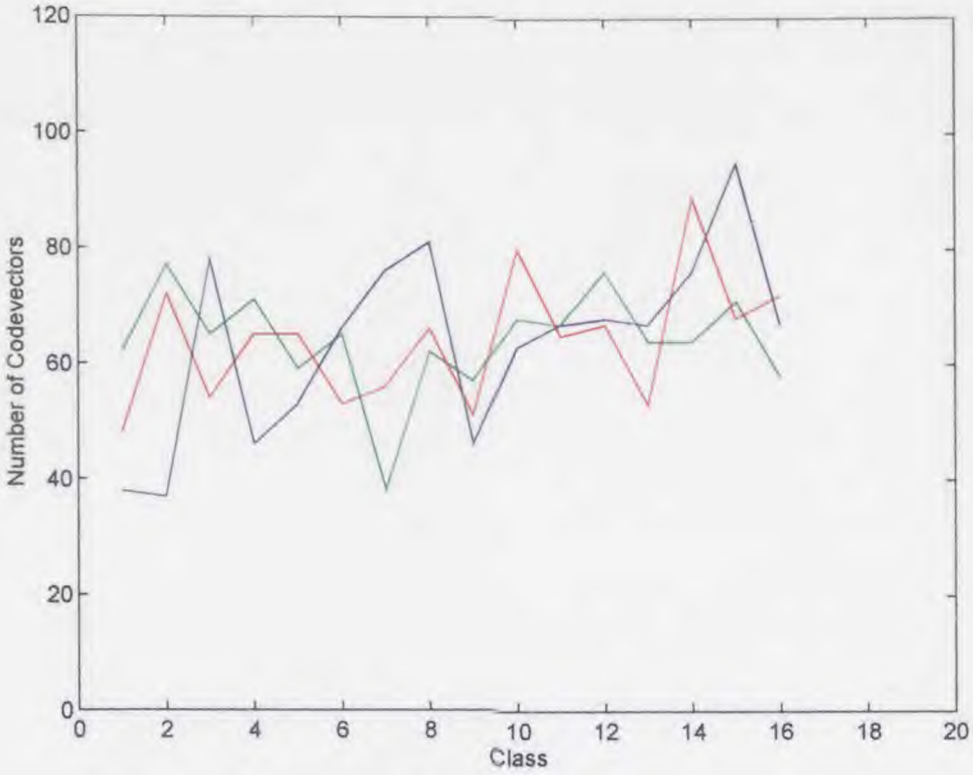

Figure 4.3 VQ-Based Classifier Histogram of Code Vector (N=16)

Table 4.1 Performance of VQ-Based Classifier, N=16

Num. of Class	Num. of Candidate	SD (dB)	Miss-coded (/20,000)	Complex (Max.)	Memory (word)
16	1	0.865	9453	1038	10448
	2	0.777	2579	1815	
	3	0.762	623	2524	
	4	0.758	130	3204	
	5	0.758	39	3907	
	6	0.757	8	4583	

Table 4.2 Performance of VQ-Based Classifier, N=32

Num. of Classes	Num. of Candidate	SD (dB)	Miss-coded (/20,000)	Complex (Max.)	Memory (word)
32	1	0.867	10746	767	10656
	2	0.784	3873	1196	
	3	0.765	1170	1580	
	4	0.759	309	1978	
	5	0.758	98	2340	
	6	0.757	28	2710	

Table 4.3 Performance of VQ-Based Classifier, N=64

Num. of Classes	Num. of Candidate	SD (dB)	Miss-coded (/20,000)	Complex (Max.)	Memory (word)
64	1	0.887	12506	876	11072
	2	0.796	5519	1077	
	3	0.770	2082	1272	
	4	0.762	732	1466	
	5	0.759	269	1669	
	6	0.758	101	1853	
	7	0.757	34	2047	

Figure 4.4 illustrates all the simulation results together, for better comparison.

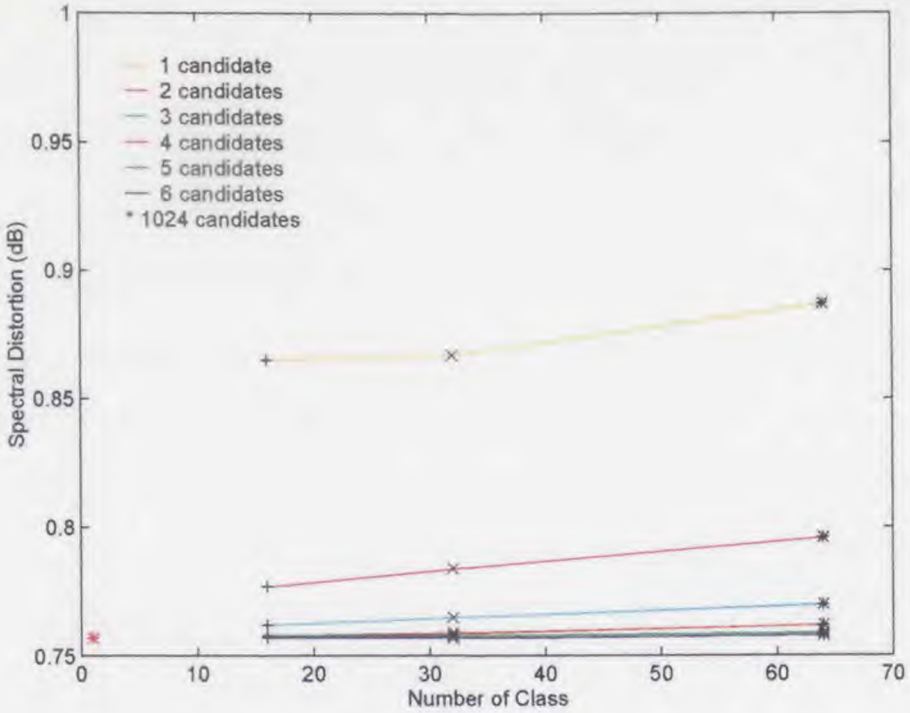

Figure 4.4 VQ-Based Classifier SD vs. Number of Class

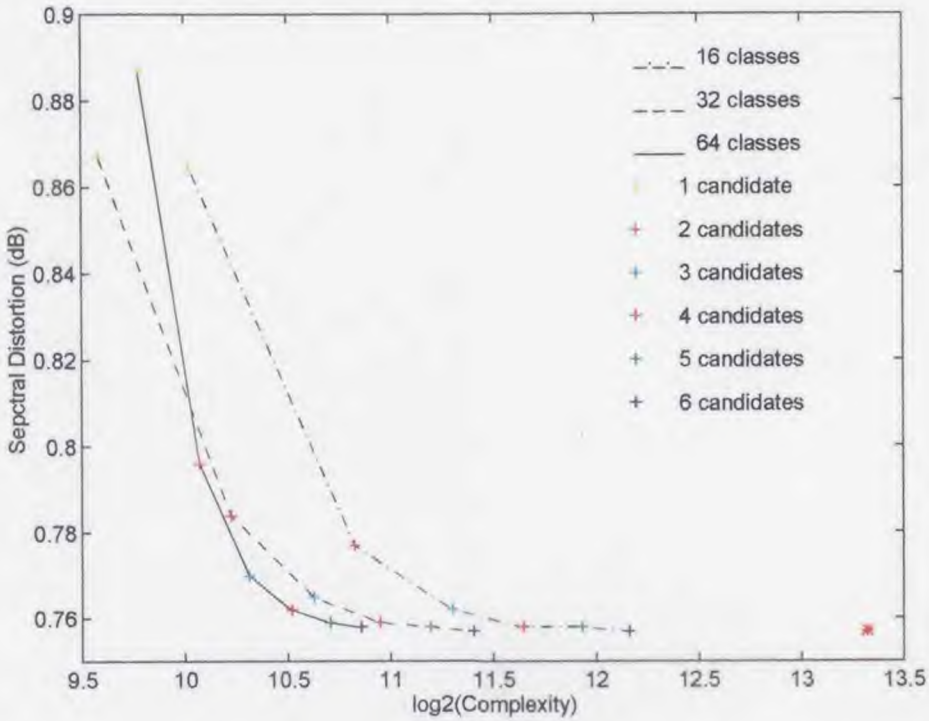

Figure 4.5 VQ-Based Classifier SD vs. Complexity

In Figure 4.4, a different color indicates a different number of candidates. In a VQ-based classifier, for a certain class size, the performances get better when more candidates are taken. For a certain number of candidates, the performances gets better when the individual codebook is partitioned into fewer classes--equivalent to a large class size. Both of the changes enlarge the search range and improve the performance. At the same time, the complexity is increased.

In Figure 4.5, the change of complexity versus spectral distortion is shown clearly. Overall, the complexity increases while the spectral distortion decreases. But this change is not linear. The point with least complexity and least spectral distortion is found in the 6-candidate, 64-class case, which should be the best compromise. The simulation results fit our expectations.

CHAPTER 5

VQ-BASED CLASSIFIER WITHOUT CLASS CODEBOOK

5.1 Definition

After the previous simulation test, the memory cost of the VQ-based classifier should be noticed. In the full-search VQ

codebook = main codebook.

In the VQ-based classifier with class codebook

codebook = index table + class codebook + main codebook.

The previous simulation shows that a 64-class case involves lower complexity than the 32-class case in order to get the optimal quantization. But the 64-class case needs twice the storage for both the class codebook and the index table. To avoid the extra storage for the class codebook, this scheme uses the existing code vector in the optimal codebook nearest to the class vector to replace the class vector itself. Namely, after designing of the class codebook, the optimal codebook is partitioned into N sets (classes) defined by N centroid vectors that are members of the optimal codebook. The codebook is rearranged such that a centroid and all its nearest neighbors occupy a contiguous segment of the codebook. An index table is used for pointing to centroids. The classifier uses this table to find m nearest centroid candidates to an input sub-vector. The corresponding sets are then searched for the final code vector.

Obviously, these replacements effects the overall performance. It is predictable that with same number of class and candidates, the VQ-based

classifier without class codebook has higher spectral distortion than that with class codebook.

The complexity of this scheme is same as the previous one, and the added memory (index table) is $3 \times N$ words. The benefit of this scheme is that for any class size the only extra storage is the index table and

codebook = index table + main codebook.

5.2 Simulation Results

Tables 5.1, 5.2, and 5.3 show the performance in the same way as Chapter 4 does. Figure 5.1 and Figure 5.2 give a better overview of the comparison.

Table 5.1 Performance of VQ-Based Classifier Without Class Codebook, N=16

Num. of Class	Num. of Candidate	SD (dB)	Miss-coded (/20,000)	Complex (Max.)	Memory (word)
16	1	0.889	10400	1038	10288
	2	0.785	3273	1815	
	3	0.764	877	2524	
	4	0.759	205	3224	
	5	0.758	55	3907	
	6	0.757	16	4601	

Table 5.2 Performance of VQ-Based Classifier Without Class Codebook, N=32

Num. of Class	Num. of Candidate	SD (dB)	Miss-coded (/20,000)	Complex (Max.)	Memory (word)
32	1	0.917	12230	767	10336
	2	0.800	5109	1181	
	3	0.770	1911	1580	
	4	0.762	740	1978	
	5	0.760	312	2340	
	6	0.758	161	2723	
	7	0.758	76	3075	
	8	0.757	33	3452	

Table 5.3 Performance of VQ-Based Classifier Without Class Codebook, N=64

Num. of Class	Num. of Candidate	SD (dB)	Miss-coded (/20,000)	Complex (Max.)	Memory (word)
64	1	0.953	14237	868	10432
	2	0.821	7450	1074	
	3	0.782	3453	1274	
	4	0.768	1610	1467	
	5	0.762	730	1666	
	6	0.760	341	1852	
	7	0.758	153	2064	
	8	0.758	70	2254	
	9	0.758	39	2418	
	10	0.757	20	2610	

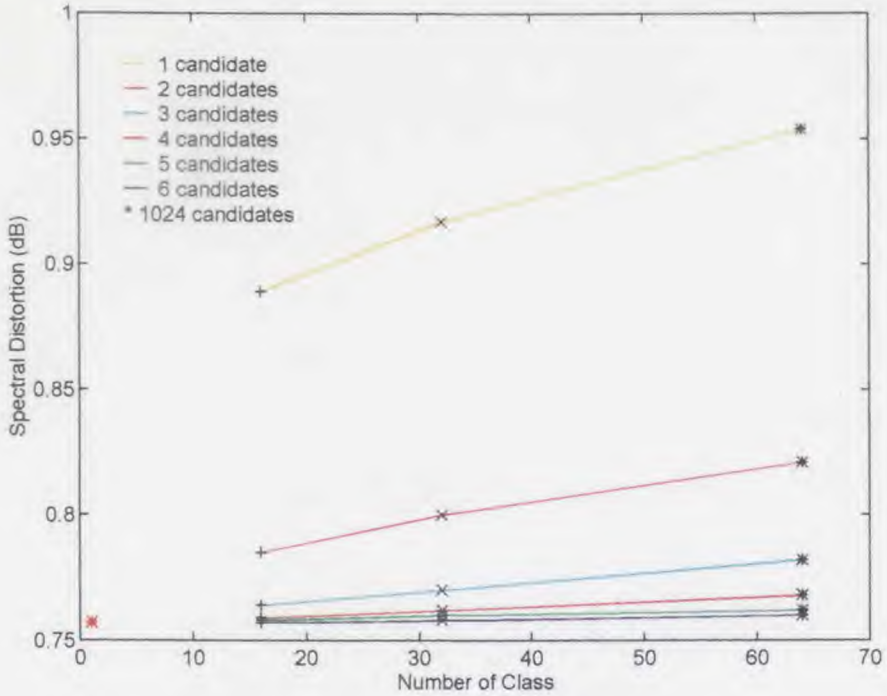

Figure 5.1 VQ-Based Classifier Without Class
Codebook SD vs. Number of Class

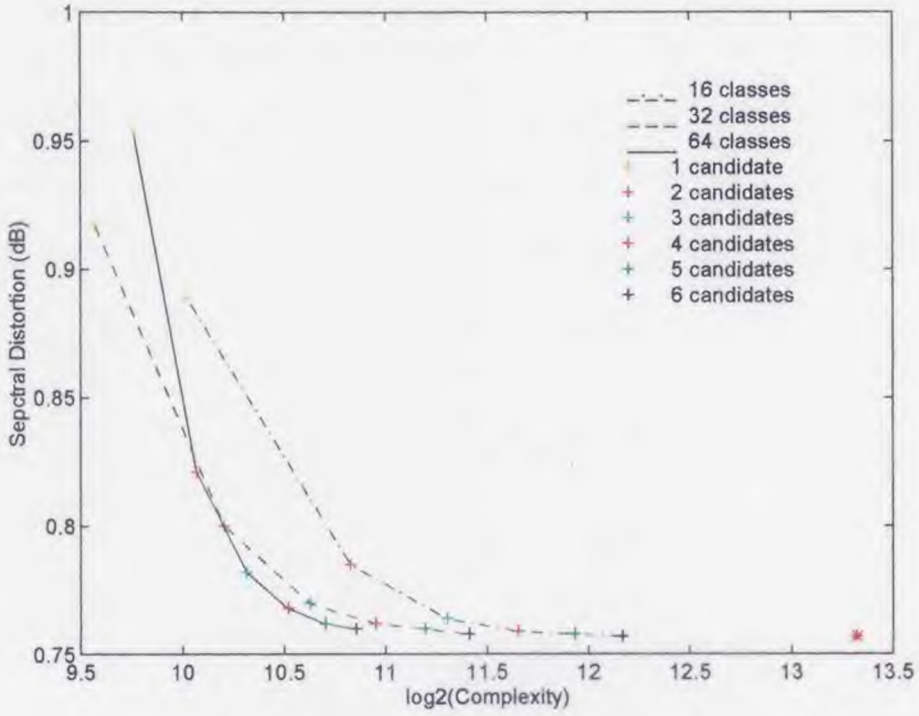

Figure 5.2 VQ-Based Classifier Without Class
Codebook SD vs. Complexity

CHAPTER 6

SQ-BASED CLASSIFIER OF LSF

Although the VQ-based classifier achieves optimal performance with lower complexity than the full-search does, it's first-stage classifier is a vector quantization and contributes considerable complexity while the number of class increases and more class vectors exist. Scalar quantization attempts to replace vector quantization as first-stage quantization for fast classification.

6.1 SQ-Based Classifier of LSF

As mentioned before, one of the benefits of using LSF to represent an LPC character is its small dynamic range$(0-\pi)$. Therefore a scalar quantizer is first used directly on the LSF components.

First, the dynamic range of each code vector component/dimension is divided into regions in such a manner that the distribution between regions is as even as possible, and each region is given a region index. Then, for a k-dimensional individual codebook, code vectors are sorted by new indices, which are the combinations of all their component's region indices.

Given a k-dimensional N output individual codebook

$$C \equiv \{y_1, y_2, y_3, \cdots, y_N\} = \begin{Bmatrix} y_{11} & y_{21} & \cdots & y_{N1} \\ y_{12} & y_{22} & \cdots & y_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ y_{1k} & y_{2k} & \cdots & y_{Nk} \end{Bmatrix} = \begin{Bmatrix} C_1 \\ C_2 \\ \vdots \\ C_k \end{Bmatrix},$$

where C_i's, for $i = 1,2,\cdots k$, are one-dimensional targets, and scalar quantization is used on them, respectively. There are k new codebooks c_i's with m_i output levels:

$$c_i \equiv \left\{ y'_{1i}, y'_{2i}, \cdots, y'_{m_i i} \right\} \subset C_i .$$

Notice that m_i's could vary from one dimension to the other, so dimensions can be partitioned into a different number of regions.

The indexing of the output values is chosen so that

$$y'_{1i} < y'_{2i} < y'_{3i} < \cdots < y'_{m_i i} .$$

For an input code vector, the k scalar quantization give k indices I_1 to I_k, and the encoding index is the combination of them, namely

$$I = \sum_{i=1}^{k} I_i \prod_{j=1}^{i} m_{j-1} ,$$

where $m_0 = 1$. The individual codebooks are then rearranged according to these indices.

The advantage of this method lies in the simple first-stage scalar quantizations whose complexities are negligible. This advantage is evident when uniform quantizations are used.

To classify a three-dimensional VQ, one need minimize the distance

$$D = (y_1 - y'_1)^2 + (y_2 - y'_2)^2 + (y_3 - y'_3)^2 .$$

To uniformly classify a three-dimensional SQ, there are 3 simple multiples:

$$y_1 : \quad I_1 = \text{int}[(y_1 - y_{10}) * \Delta_1] + 1$$

$$y_2 : \quad I_2 = \text{int}[(y_2 - y_{20}) * \Delta_2] + 1$$

$$y_3: \quad I_3 = \text{int}[(y_3 - y_{30}) * \Delta_3] + 1,$$

where y_{i0}s and Δ_is are starting points and steps.

The class index is:

$$I = I_1 + m_1 * I_2 + m_1 * m_2 * I_3$$

Going through 3 or 4 scalar quantizers simultaneously, the encode searching is restricted in a range much smaller than the whole codebook. The storage cost of this scheme is the scalar quantization codebooks c_is. This cost can be reduced by using uniform scalar quantization, so for each dimension only the starting point and step need to be saved. So

codebook = index table + 10 * (step + starting point) + main codebook

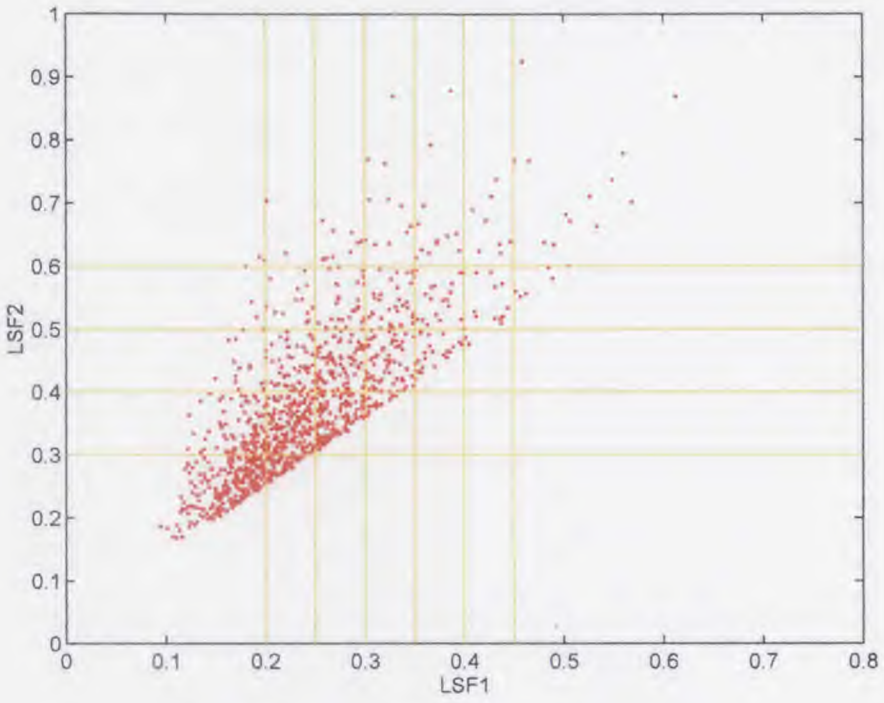

Figure 6.1 SQ-Based Classifier on Each Component

This scheme quantizers a k-dimensional input with k random variables, each quantized by a scalar quantizer. Figure 6.1 shows the two-

dimensional resulting vector quantizer corresponding to a particular choice of scalar quantization for each variable. It is evident that the VQ defined by separately quantizing the components of a vector must always result in quantization cells that are rectangular. In contrast, a more general vector quantizer is freed from these geometrical restrictions and can have arbitrary cell shapes. In higher dimensions the same idea is clearly applicable. Thus, in three dimensions, scalar quantization of the three components of a vector always results in cells that have rectangular, box-like shapes, where each face is a plane parallel to one of the coordinate axes. On the other hand, regular quantizers in three dimensions will have polyhedral cells. Extending this idea to k dimensions, it is clear that scalar quantization of the components of a vector always generates a very restricted class of vector quantizers, where the faces are (k-1)-dimensional hyperplanes--each parallel to a coordinate axis in the k-dimensional space. The inherent superiority of VQ is thereby evident simply because of the greater structural freedom it allows in quantization of a vector. It is very hard to have an equal or an almost equal number of code vectors between classes. Another disadvantage is that one needs to search more than one candidate for optimal quantization, because for each dimension, there are two possible closest candidates. So for three-dimensional sub-vector there are 8 candidates and there are 16 for four-dimensional case. The consequence is sub-optimal quantization performance with low complexity.

6.2 SQ-Based Classifier of Difference

6.2.1 Definition

Studying of Figure 6.1, a new classifier based on the differences between adjacent LSFs is attempted. Because the LSF differences are observed to be less divergent than the absolute frequencies themselves, this method is expected to give a better performance than the first one. For a three-dimensional individual codebook with code vector $[X_1, X_2, X_3]$, the scalar quantization is taken on $[X_1, \Delta X_1, \Delta X_2]$, where

$$\Delta X_1 = X_2 - X_1,$$

$$\Delta X_2 = X_3 - X_2.$$

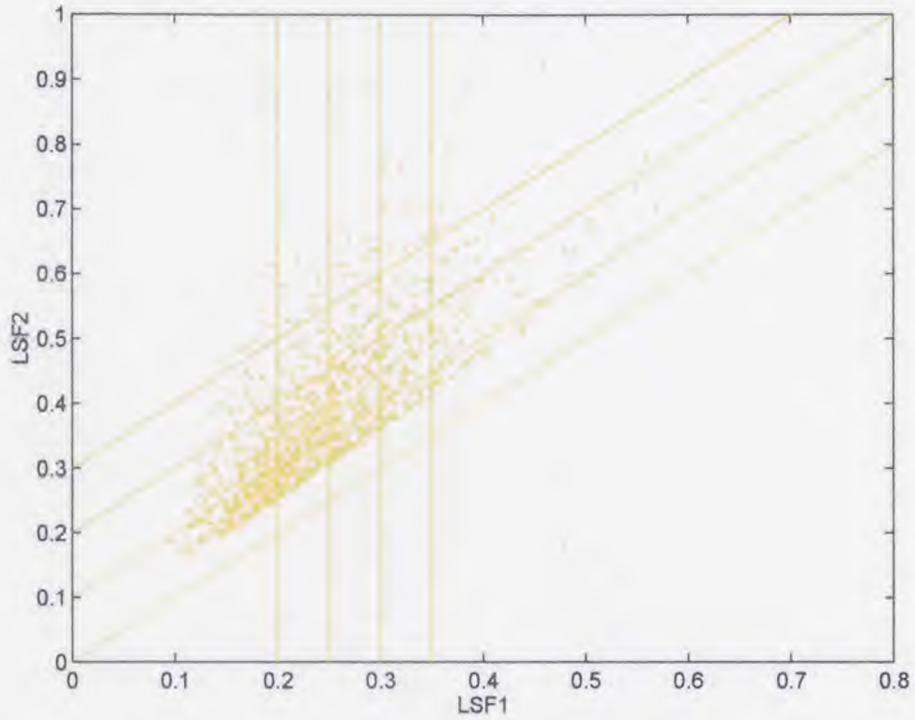

Figure 6.2 SQ-Based Classifier on Difference

Figure 6.2 illustrates a two-dimensional case of this classifier. As one may find from this figure, there is some improvement, but the codewords are still hard to be placed evenly between classes, which is proved in the simulation test. Meanwhile the problem of more candidates still exists.

By using uniform scalar quantization, the complexity of first-stage is negligible. The complexity of this scheme is roughly $\frac{10 \times 1024}{N} \times m$ and added memory includes the index table, the start value and the steps. If one dimension is divided to two classes, instead of start value and steps, only one middle boundary needs to be stored.

6.2.2 Simulation Results

Figure 6.3 is the histogram of code vectors between classes (N=16). Tables 6.1, 6.2, and 6.3 show the performance of 16-, 32-, and 64-class cases, respectively. Figures 6.4 and 6.5 illustrate the comparison for different number of class.

Table 6.1 Performance of SQ-Based Classifier, N=16

Num. of Classes	Num. of Candidate	SD (dB)	Miss-coded (/20,000)	Complex (Max.)	Memory (word)
16	1	0.840	9443	1256	10300
	2	0.800	5654	2079	
	3	0.781	3417	3089	
	4	0.767	1407	3905	
	5	0.767	1407	4912	
	8/8/16	0.759	360	8619	

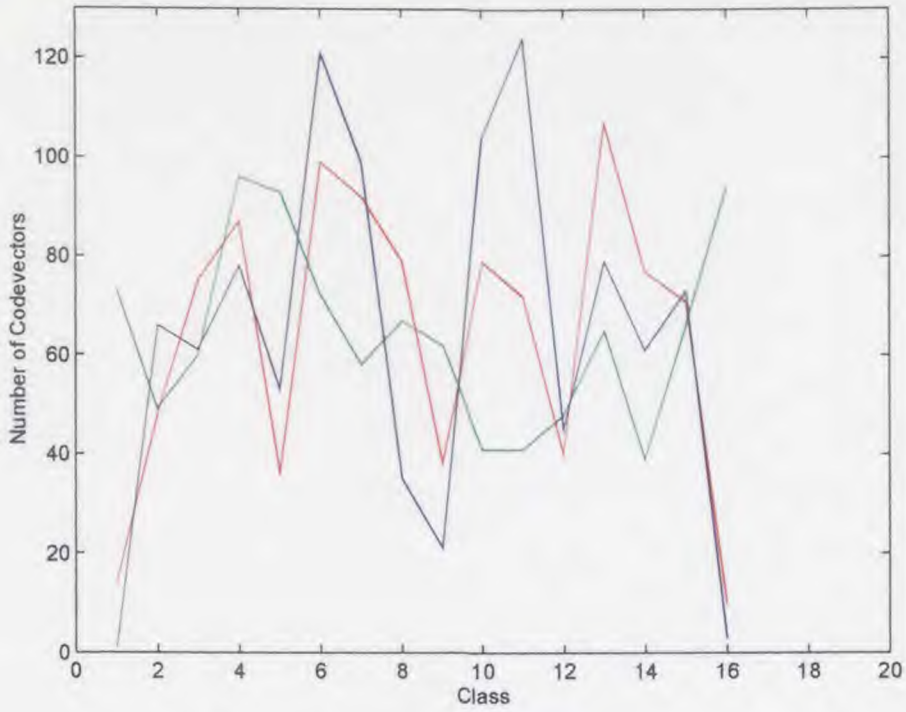

Figure 6.3 SQ-Based Classifier Histogram of Code Vector

Table 6.2 Performance of SQ-Based Classifier, N=32

Num. of Class	Num. of Candidate	SD (dB)	Miss-coded (/20,000)	Complex (Max.)	Memory (word)
32	1	0.886	12007	741	10351
	2	0.819	7915	1367	
	3	0.784	3934	1996	
	4	0.771	2140	2345	
	5	0.771	2140	1940	
	8/8/16	0.767	1584	5250	

Table 6.3 Performance of SQ-Based Classifier, N=64

Num. of Class	Num. of Candidate	SD (dB)	Miss-coded (/20,000)	Complex (Max.)	Memory (word)
64	1	0.961	13736	528	10450
	2	0.862	10535	953	
	3	0.802	6267	1277	
	4	0.780	3308	1438	
	5	0.780	3308	1762	
	8/8/16	0.774	2327	3119	

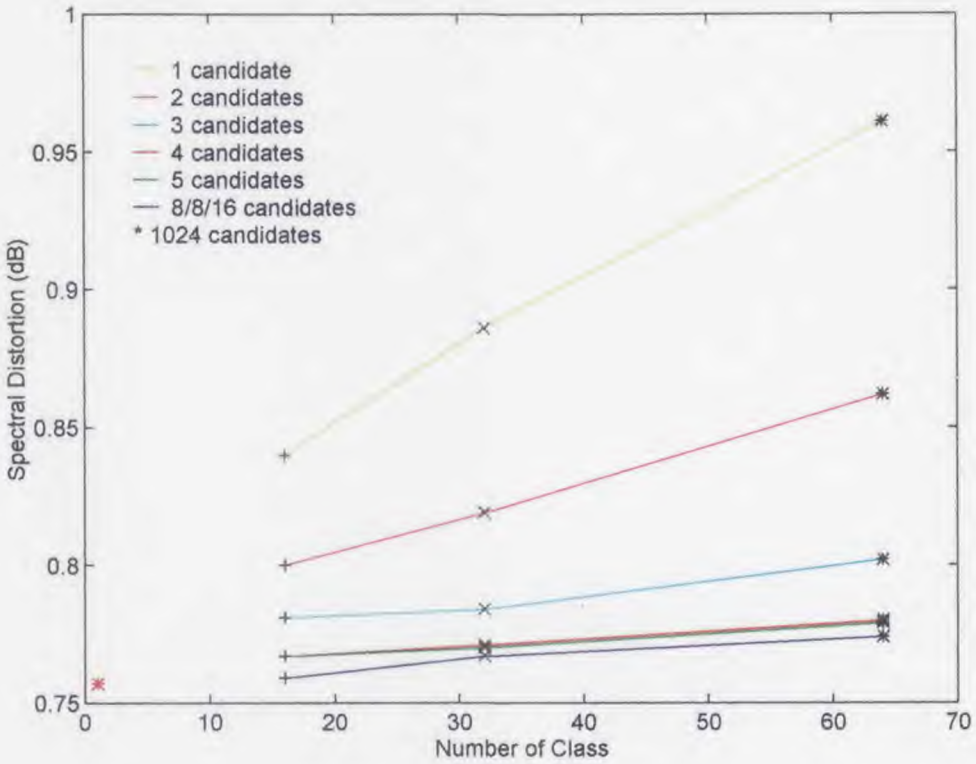

Figure 6.4 SQ-Based Classifier SD vs. Number of Class

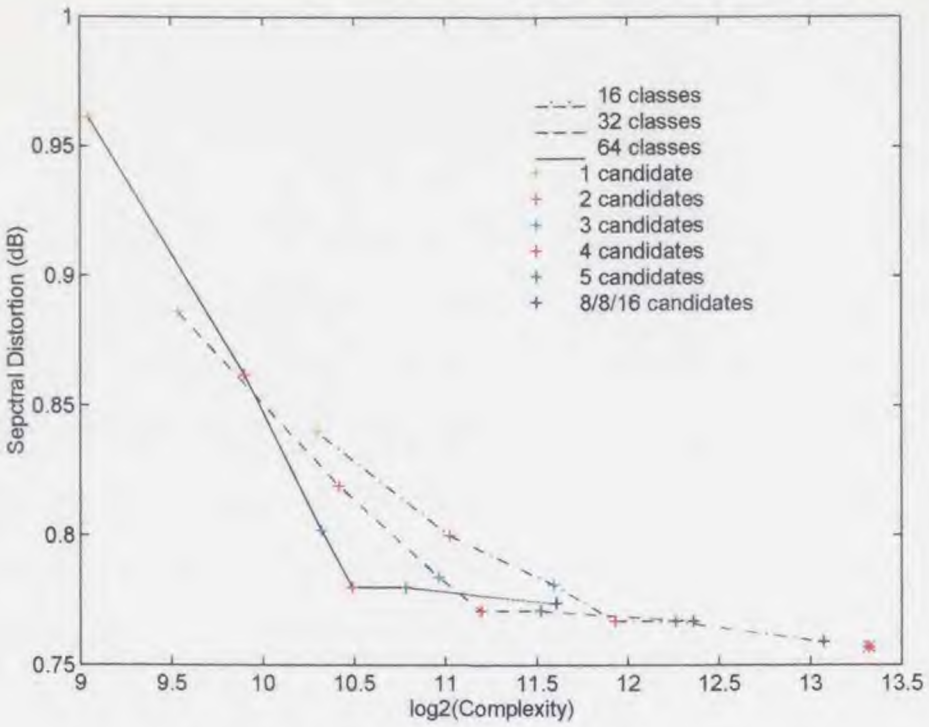

Figure 6.5 SQ-Based Classifier SD vs. Complexity

CHAPTER 7

SORTED CODEBOOK VQ

After performing all the above tests, we find that although scalar quantization itself supplies low complexity, using it on the individual components of LSFs doesn't give good results. Vector quantization gives an overview to the sub-vectors, so it has lower spectral distortion, but as a price, the complexity is higher. An effort has been made to combine the benefits and get rid of the shortages of both methods. The new classifier should not only give an overview, but it is also very simple. To be simple, the classify parameter should be fit for scalar quantization--a one-dimensional component. For an overview, it should contain all the information the code vector has. That means that we should use one number to represent the character of each code vector. A mean of the code vectors meets these requirements. It is an approximation to the gain of the code vector. The gain is the root mean-square value of the vector components and serves as a normalizing scale factor. It is defined as

$$g = \|Y\| = \sqrt{\sum_{i=1}^{k} y_i^2} \,.$$

Obviously, the mean requires much lower complexity as square calculation.

7.1 Definition

Given a k-dimensional target vector $[X_1, X_2, \cdots, X_k]$ and a codebook C of size N, we define a sorting parameter $s_t = g(X_1, X_2, \cdots, X_k)$, which is a scalar by definition, where $g(\cdot)$ is a suitable function, chosen in such a way that

neighboring target vectors give neighboring values of s_t. Then the indices of the codebook are sorted in ascending order of the sorting parameter for each code vector, according to the vector $S=[s_1, s_2, \cdots, s_N]$ with $s_1 \le s_2 \le \cdots \le s_N$.

In this case, the chosen sorting parameter s_t is simply the sum of the components in each sub-vector. The codebook is geometrically partitioned into classes by parallel hyperplanes as illustrated in Figure 7.1 for a two-dimensional case.

To accomplish this, the codebook is rearranged in an order of increasing code vector means and then divided into N equal-sized sets. N means of the 1st vector in each set are held in a class codebook.

Unlike the previous schemes, once this is done, no change in the codebook structure is necessary. The even distribution of code vectors between classes is not a problem anymore. For the individual codebook partition into N classes, every $\frac{1024}{N}$ code vectors are in same class. If 1024 is not divisible by n, then each class except the last one has $\left[\frac{1024}{N}\right]$ code vectors, where [ ] is an integer calculation. The last class has the rest of the code vectors. The first code vector mean in each class is stored as the class codebooks.

To code the target vector, the classifier extracts the mean of the input vector and performs scalar quantization using the class codebook. Assume s_i is the result of scalar quantization, with $1 \le i \le N$. The index of s_i is called the central index. In the next step, the target vector is vector quantized using an extensive local search in the neighborhood of the central index. For

example, only the code vectors with indices within the range of $i-p+1$ to $i+p$ may be searched, where p is an offset value. The offset value for the final codebook search is varied by the number of class.

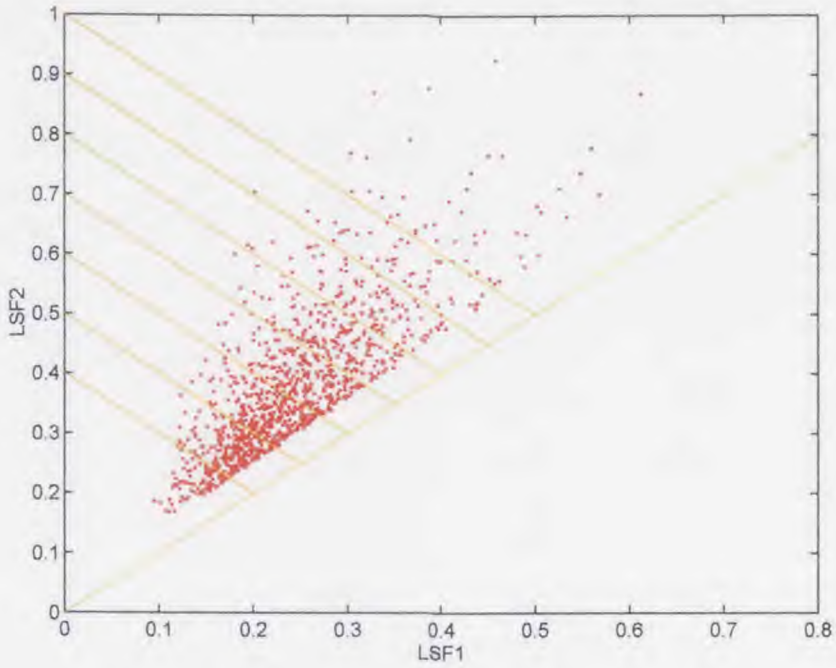

Figure 7.1 Boundary of Sorted Codebook VQ

Except the simple first-stage classification by scalar quantization, this scheme can assign each class exactly the same number of code vectors, which not only guarantees better average performance, but it also has the benefit that no index table is necessary. The new codebook structure is

codebook = class codebook + main codebook,

and the added memory is $3\times N$ words. The complexity of this scheme is $3\times\log(N)+\frac{10\times 1024}{N}\times m$, where the first and second terms are for the classifier and the quantizer, respectively.

7.2 Simulation Results

Like the previous schemes, this test simulates 16-, 32-, and 64-class and different candidate conditions, respectively. Tables 7.1-7.3 present the results and Figures 7.2 - 7.3 illustrate the comparison between those tests. No histogram figures are shown because of the exactly even distribution.

Table 7.1 Performance of Sorted Codebook VQ, N=16

Num. of Class	Num. of Candidate	SD (dB)	Miss-coded (/20,000)	Complex (average)	Memory (word)
16	1	0.903	12549	652	10288
	2	0.773	2387	1292	
	3	0.759	372	1932	
	4	0.758	73	2572	
	5	0.757	21	3212	

Table 7.2 Performance of Sorted Codebook VQ, N=32

Num. of Class	Num. of Candidate	SD (dB)	Miss-coded (/20,000)	Complex (average)	Memory (word)
32	1	1.113	17301	335	10336
	2	0.863	9456	655	
	3	0.793	4230	975	
	4	0.769	1702	1295	
	5	0.761	661	1615	
	6	0.759	227	1935	
	7	0.758	100	2255	
	8	0.758	65	2575	
	9	0.757	31	2895	

Table 7.3 Performance of Sorted Codebook VQ, N=64

Num. of Class	Num. of Candidate	SD (dB)	Miss-coded (/20,000)	Complex (average)	Memory (word)
64	1	1.505	19364	178	10432
	2	1.102	16616	338	
	3	0.939	12686	498	
	4	0.857	8593	658	
	5	0.816	5897	818	
	6	0.791	3795	978	
	7	0.777	2551	1138	
	8	0.768	1560	1298	
	9	0.763	991	1458	
	10	0.760	554	1618	
	11	0.759	367	1778	
	12	0.759	203	1938	
	13	0.758	138	2098	
	14	0.758	92	2258	
	15	0.758	79	2418	
	16	0.758	53	2578	
	17	0.757	39	2738	

For optimal results, the sorted codebook VQ uses only a 26% search complexity as the full-search VQ and the storage cost is increased only 1.8%.

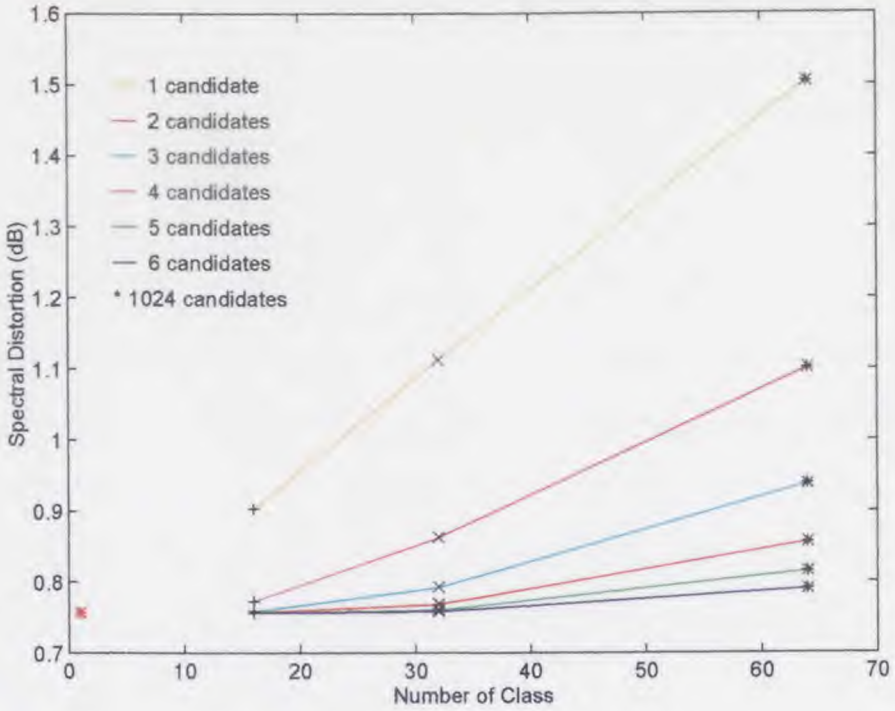

Figure 7.2 Sorted Codebook VQ SD vs. Number of Class

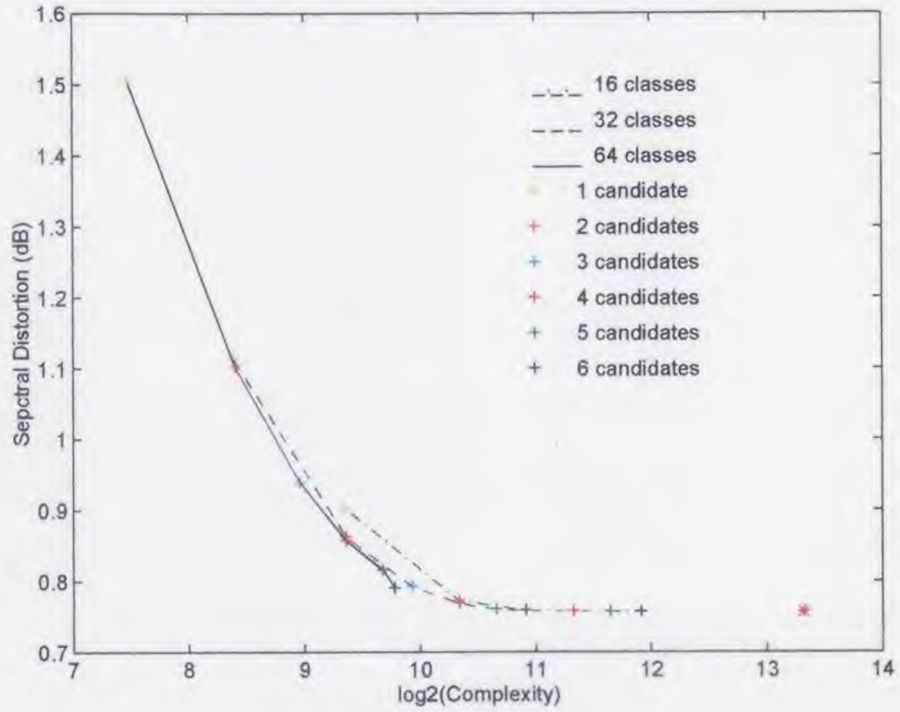

Figure 7.3 Sorted Codebook VQ SD vs. Complexity

CHAPTER 8

CONCLUSION

In this thesis we have designed different classifiers based on split vector quantization. Employing the classifier to the VQ codebook reconstructs the codebook and gathers the code vectors with the same characters together, making the coding search efficient and simple. Quantization performance results demonstrate that optimal performance, the same as with unconstrained VQ, can be obtained from most of these methods. The major benefit of these classifications is that considerable complexity can be saved for vector quantization while maintaining comparable performance.

Table 8.1 Comparison of Four Schemes

Schemes	SD (dB)	Miss-coded (/20,000)	Complex (Average)	Memory (word)
VQ with Class Codebook	0.757	34	2047	11072
VQ without Class Codebook	0.757	20	2610	10432
SQ of Differences	0.759	360	8619	10300
Sorted Codebook VQ	0.757	39	2738	10432

Table 8.1 summarizes the performance of all the CVQ systems tested. It is shown that an average distortion of a full-search VQ can be achieved by low-complexity CVQ at only about 25% of the full-search complexity and with a very small miss-coding count. This is accomplished by the VQ and the sorted codebook VQ classifier. Figure 8.1 shows the distortion versus

complexity curves for the tested systems. The use of these schemes for low-complexity speech coding depends on the best trade-off for the application in mind.

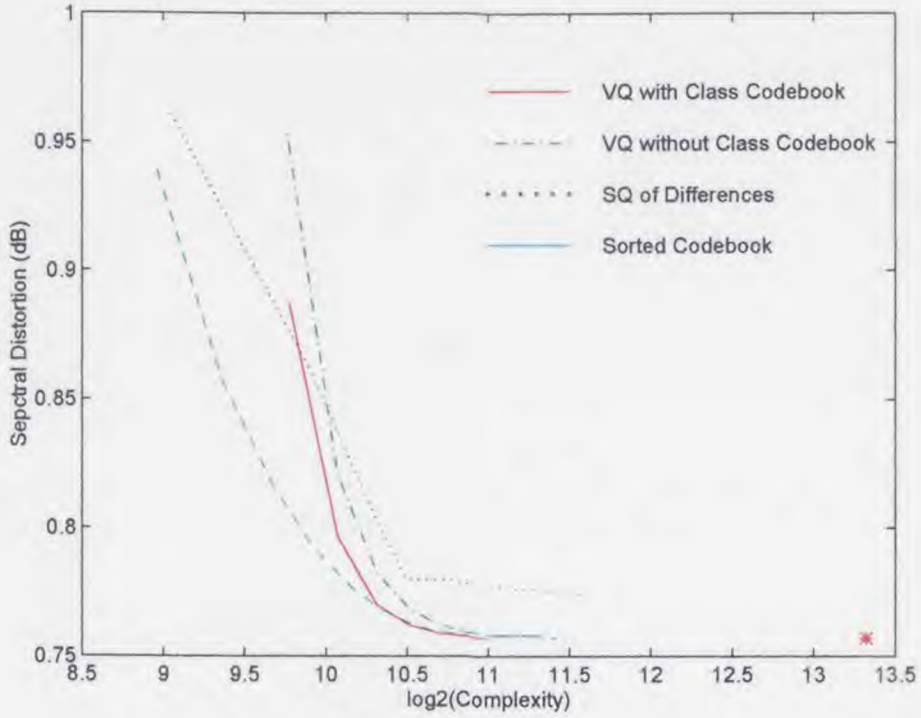

Figure 8.1 Spectral Distortion vs. Complexity

REFERENCES

1. F. Itakura, "Linear spectrum representation of linear predictive coefficients of speech signals," *J. Acoust. Soc. Amer.*, vol. 57, suppl. no. 1, pp. 35, 1975

2. F. K. Soong and B. H. Juang, "Line Spectrum Pair (LSP) and speech data compression," *ICASSP*, pp. 1.10.1, 1984.

3. K. K. Paliwal and B. S. Atal, "Efficient vector quantization of LPC parameters at 24 bits/frame," *IEEE Transactions on Speech and Audio Processing*, pp. 3-7, Jan., 1993.

4. B. Ramamurthi and A. Gersho, "Classified vector quantization of images," *IEEE Transactions on Communications*, pp.1105-1109, Nov., 1986.

5. F. K. Soong and B. H. Juang, "Optimal quantization of LSP parameters", *IEEE Transactions on Speech and Audio Processing*, pp. 15-19, Jan., 1993.

6. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Norwell, MA, 1991.

7. J. D. Markel and A. H. Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag, Berlin, Heidelberg, New York, 1976.

8. L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ, 1978.

9. R. Hagen, *On Robust LPC-spectrum Coding and Vector Quantization*, Chalmers University of Technology, Goteborg, Sweden, 1995.

10. H. R. Sadegh Mohammadi and W. H. Holmes, "Low cost vector quantization methods for spectral coding in low rate speech coders," *ICASSP*, pp.720-723, 1995.

11. H. R. Sadegh Mohammadi and W. H. Holmes, "Application of sorted codebook vector quantization to spectral coding of speech," *Globecom*, pp.1595-1598, 1995.