Waveform Coding Techniques based on Vector Quantization

A thesis submitted
for award of the Degree of

Doctor of Philosophy in
Electrical Engineering

Submitted By:
Arun Nandurbarkar

Guided By:
Prof. S. M. Joshi

ELECTRICAL ENGINEERING DEPARTMENT
FACULTY OF TECHNOLOGY & ENGINEERING
THE MAHARAJA SAYAJIRAO UNIVERSITY OF BARODA
VADODARA – 390 001, GUJARAT, INDIA.

November 2011

Dedicated

To

GOD

Certificate

This is to certify that the thesis entitled, “Waveform Coding Techniques based on Vector Quantization” submitted by **Shri Arun Nandurbarkar** in fulfillment of the degree of **DOCTOR OF PHILOSOPHY** in Electrical Engineering Department, Faculty of Technology & Engineering, The M. S. University of Baroda, Vadodara is a bonafide record of investigations carried out by him in the Department of Electrical Engineering, Faculty of Technology & Engineering, M. S. University of Baroda, Vadodara under my guidance and supervision. In my opinion, this has attained the standard fulfilling the requirements of the Ph.D. Degree as prescribed in the regulations of the University.

November, 2011

Prof. S. M. Joshi
Department of Electrical Engineering,
Faculty of Technology & Engineering,
The Maharaja Sayajirao University of Baroda,
Vadodara – 390 001

Declaration

I, **Shri Arun Nandurbarkar** hereby declare that the work reported in this thesis entitled, "**Waveform Coding Techniques based on Vector Quantization**" submitted for the award of the degree of **DOCTOR OF PHILOSOPHY** in Electrical Engineering Department, Faculty of Technology & Engineering, The M. S. University of Baroda, Vadodara is original and was carried out in the Department of Electrical Engineering, Faculty of Technology & Engineering, M. S. University of Baroda, Vadodara. I further declare that this thesis is not substantially the same as one, which has already been submitted in part or in full for the award of any degree or academic qualification of this University or any other Institution or examining body in India or abroad.

November, 2011 **Shri Arun Nandurbarkar**

Acknowledgements

I express my thanks to **Almighty** for providing me inspiration, strength, energy and patience to start and accomplish my goal with the support of all concerned , a few of them I am trying to acknowledge.

I sincerely thank my guide **Prof. S. M. Joshi**,who has been the main source of inspiration, for guiding and motivating me every time I felt nervous. His sincerity, dedication inspired me to work tirelessly for the entire duration of this work.

I thank **Prof. A.I.Trivedi, Prof. Smt S.A. Kanitkar and Prof. S.K.Shah**, Electrical Engineering Department, Faculty of Technology & Engineering, M. S. University of Baroda for helping me in all the ways throughout the research work.

I would like to thank **Prof. M. V. Garach**, **Prof. J. V. Dave**, **and Prof. Smt. Usha Neelkanthan** for always encouraging the academic endeavors.

I express my thanks to **Kiran, Mihir, Kishor** and also **Navneet Ghedia ,M V Makwana and Smt. P J Brahmbhatt** who helped me in this research work. Finally I express my thanks to my wife **Dipali** for all the support, inspiration and love given to me despite all inconveniencies caused because of my engagement in this work.

November, 2011 **Arun Nandurbarkar**

Abstract

Compression squeezes data so it requires less storage space and less bandwidth for transmission. Compression is usually used for speech, image and video applications.

The compression techniques can be classified as lossy and lossless compression. With lossy compression, it is assumed that some loss of information is acceptable. Loss is also acceptable in voice and audio compression, depending on the desired quality. The removal of information in the lossy technique is acceptable for images, because the loss of information is imperceptible to the human eye in some cases. In lossless compression, data is compressed without any loss of data.

Lossy compression can be achieved by quantizing a sample with less number of bits.Quantization can be classified into scalar and vector quantization. Scalar Quantization (SQ) involves processing the input samples individually. An SQ involves mapping of each individual input to an output using some distortion measure. Scalar Quantization is used, primarily for analog to digital conversion. With Vector Quantization (VQ) processing of the input samples is carried out in group of samples. The input is divided into pieces called vector. Essential to this type of encoding is the presence of a `codebook', an array of vectors. For each vector of the input, the closest match to a vector in the codebook is looked up. To find the closest match some distortion measure is used. The index of this codebook entry is then used to encode the input vector.

Major contributions of this thesis are: comprehensive study of concept of vector quantization , review of waveform coding, design and performance of unconstrained vector quantizers , design and performance of constrained vector quantization techniques and comparison of various techniques.

The major results obtained summarized here. Code books with uniform pdf have been applied to vector quantization techniques. For the first time code books with Gaussian pdf have been applied to vector quantization techniques over here.

Code books with Gaussian pdf compared to uniform pdf give better performance in case where Full Search VQ (FSVQ) is applied to speech. SQNR 3.5707 dB more is be obtained in case of speech.

As the number of code words is increased SQNR and hence PSNR also gets better in case where Full Search VQ is applied to an image. In case where FSVQ is applied to image, code books with Gaussian pdf compared to uniform pdf does not produce better results. This indicates the random distribution of pixel amplitudes in the image.

As the tree depth increases the SQNR gets better in case of Tree Structured VQ (TSVQ) applied to speech. Code books with Gaussian pdf compared to uniform pdf gives generally better performance in case where TSVQ is applied to speech. SQNR improvement of maximum 0.102 dB is obtained.

Code books with Gaussian pdf compared to uniform pdf gives better performance in case where TSVQ is applied to image. PSNR improvement of maximum 6.981 dB is obtained.

With Multi Stage VQ (MSVQ) is applied to speech, maximum SQNR improvement of 4.732, 12.867 and 23.6025 dB is obtained for the three different speech signals respectively.

Code books with Gaussian pdf compared to uniform pdf give better performance in case where Trellis Coded VQ (TCVQ) is applied to speech. Performance improvement in SQNR from 4-state trellis to 8-state trellis is maximum 0.0984 dB (block-delayed), 0.0058 dB (viterbi) for Gaussian pdf.

CONTENTS

List of Figures

List of Tables

List of Abbreviations

Bpp	Bits Per Pixels
CR	Compression Ratio
CVQ	Classified Vector Quantization
CWT	Continuous Wavelet Transform
DCT	Discrete Cosine Transform
DSP	Digital Signal Processing
DWT	Discrete Wavelet Transform
FSVQ	Full-search Vector Quantization
GIF	Graphic Interchange Format
GLA	Generalized Lloyd Algorithm
LUT	Look-Up Table
MSE	Mean Square Error
MSVQ	Multi Stage Vector Quantization
NN	Nearest Neighbor
pdf	Probability Density Function
PNN	Pairwise Nearest Neighbor
PSNR	Peak Signal to Noise Ratio
RLE	Run Length Encoding
SQ	Scalar Quantization
SQNR	Signal to Quantization Noise Ratio
STFT	Short Time Fourier Transforms
SR-TVQ	Shift Register Trellis Vector Quantization
TCVQ	Trellis-code Vector Quantization
TSVQ	Tree-Structured Vector Quantization
TIFF	Tagged Image File Format
VLSI	Very Large Scale Integration
VQ	Vector Quantization

Arun Nandurbarkar
List Of Papers Published in International / National Conference Proceedings

[1] A.B.Nandurbarkar, Prof S M Joshi, and P I Panchal , "Linear Predictive Coding using VQ Techniques," Conference on Smart Computing & Communications SCAC-2007, Vadodara Organized by IETE, Vadodara & Electrical Egg. Dept., Faculty of Tech. & Egg., M.S. University, Vadodara, Jan 28, 2007.

[2] A.B.Nandurbarkar, Prof S M Joshi, and M V Makwana, "Tree-Structured Vector Quantization-An Overview," Conference on Smart Computing & Communications SCAC-2007, Vadodara Organized by IETE, Vadodara & Electrical Egg. Dept., Faculty of Tech. & Egg., M.S. University, Vadodara, Jan 28, 2007.

[3] M V Makwana, A.B. Nandurbarkar, and Prof S M Joshi, "Image compression using Tree-Structured Vector Quantization with Compact Codebook," Proceedings of International conference on "Computational & Intelligence Multimedia Applications 2007(ICCIMA'07), pp. 102-107, Organized by Mepco Schlenk Engineering College, Sivakasi, Tamilnadu , Dec 13-15 2007.

[4] M V Makwana, P I Panchal, A. B. Nandurbarkar, and Prof S M Joshi, "Speech Compression using Vector Quantization," Proceedings of National conference on Advance Communication , pp. 86-89 , Organized by DDIT , Nadiad , March 29-30 2008.

[5] A. B. Nandurbarkar and Prof S M Joshi, "Multistage Vector Quantization with Gaussian Codebook Applied to Speech," Proceedings of 3rd National

Conference on current Trends in Technology, pp. 378-380, Organized by Institute Of Technology , Nirma University , Ahmedabad , Nov 27-29 , 2008.

[6] M. V. Makwana, A. B. Nandurbarkar, and Prof. K.R. Parmar,"Vector Quantization: A Unified Approach Towards Data Compression," Proceedings of National Conference on Innovations & Applications of Mathematical Modeling Techniques in Engineering Systems, pp. 34, Organized by A.D. Patel Institute of Technology , V.V.Nagar, Dec 10, 2008.

[7] Smt. P. J. Brahmbhatt, A. B. Nandurbarkar, and Prof S M Joshi, "Vector Quantization Techniques: an overview," *Proceedings* of National Conference on emerging Trends in Communication, pp. 23, Organized by Swami Vivekanand Institute of Engg. & Tech., Chandigadh, Feb 20-21, 2009.

[8] Smt. P. J. Brahmbhatt ,A. B. Nandurbarkar, and Prof S M Joshi, "Design And Performance Of Trellis Coded Vector Quantizer for Speech Signal With Uniform And Gaussian Sources," Proceedings of National Conference on Emerging Trends in Information , Electronics & Communication, pp. 74-77, Organized by Parul Institute of Engineering & Technology , Ta. Limda, Vadodara, September 11-12,2009.

Chapter 1
Introduction

Compression squeezes data so it requires less storage space and less bandwidth for transmission. Compression is usually used for speech, image and video applications.

The compression techniques can be classified as lossy and lossless compression. With lossy compression, it is assumed that some loss of information is acceptable. Loss is also acceptable in voice and audio compression, depending on the desired quality. The removal of information in the lossy technique is sometimes acceptable for images, because the loss of information is imperceptible to the human eye in some cases. In lossless compression, data is compressed without any loss of data.

Lossy compression can be achieved by quantizing a sample with less number of bits .Quantization can be classified into scalar and vector quantization. Scalar Quantization (SQ) involves processing the input samples individually. An SQ involves mapping of each individual input to an output using some distortion measure. Scalar Quantization is used, primarily for analog to digital conversion. With Vector Quantization (VQ) processing of the input samples is carried out in group of samples. The input is divided into pieces called vector. Essential to this type of encoding is the presence of a `codebook', an array of vectors. For each vector of the input, the closest match to a vector in the codebook is looked up. To find the closest match some distortion measure is used. The index of this codebook entry is then used to encode the input vector.

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. VQ is usually, but not exclusively, used for the purpose of data compression. Nevertheless, there are interesting parallels with scalar quantization and many of the structural models used in VQ are natural generalizations of the scalar case.

A vector can be used to describe almost any type of pattern, such as a segment of a speech waveform or an image, simply by forming a vector of samples from the waveform or image. Another example, of importance in speech processing, arises when a set of parameters (forming a vector) is used to represent the spectral envelope of a speech sound. Vector quantization can be viewed as a form of pattern recognition where an input pattern is "approximated" by one of a predetermined set of standard patterns, or in other language, the input pattern is matched with one of a stored set of templates or code words. Vector quantization can also be viewed as a front end to a variety of complicated signal processing tasks, including classification and linear transforming. In such applications VQ can be viewed as a complexity reducing technique because the reduction in bits can simplify the subsequent computations, sometimes permitting complicated digital signal processing to be replaced by simple table lookups. Thus VQ is far more than a formal generalization of scalar quantization. In the last few years it has become an important technique in speech recognition as well as in speech and image compression, and its importance and application are growing.

1.1 Motivation

The purpose of quantization is to provide a limited-precision description of a previously unknown input value. It is only because the input is not known in advance that it is necessary to quantize. Thus the input must be modeled as a random variable, having some specific statistical characteristics, usually specified by its probability density function (pdf).

The work in this area has never been done with the codebook considering the pdf. This is the primary motivation for carrying out this work. Work done by others in this area earlier is described briefly below.

A technique for random coding in VQ has often been used in pattern recognition literature and was used in original development of k- means technique (1).In random coding code book is filled with code words selected randomly. Shannon source coding theorems imply that such a random selection of code words will on average yield a good code (2), (3), (4).

For designing VQ pruning is another technique which is well known in statistical clustering literature (5). Pruning refers to the idea of starting with the training set and selectively eliminating (pruning) training vectors as candidate code vectors until a final set of training vectors remain as the code book.

A more complicated, but better, means of finding a codebook from a training sequence is the pair wise nearest neighbor (PNN) clustering algorithm proposed by Equitz (6)(7). In this algorithm, best pair of clusters from training sequence is merged until the desired number of code vectors is achieved. Similar algorithms have also been used in clustering literature (8)(9).

Linde et.al. introduced a technique that resembles the product code initialization in that it grows large codebooks from small ones, but differs in that it does not require an integral no. of bits / symbol (10). This technique is called splitting. In this method, globally optimal code book of a training sequence is found, which is the centroid, $\mathbf{y_0}$, of the entire sequence. $\mathbf{y_0}$ can be split into two code words , $\mathbf{y_0}$ and $\mathbf{y_0} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a vector of small Euclidean norm. One continues in this manner until desired number of code words is obtained.

The Generalized Lloyd Algorithm (GLA) for VQ design is sometimes known as the k-means algorithm after MacQueen (1) who studied it as a statistical clustering procedure. The idea has been described earlier by Forgey in a clustering context. It has since led to a variety of extensions and applications in statistical literature, Diday & Simon (11). It is sometimes referred to as LBG algorithm in data compression literature after (10), as it was proposed by Linde , Buzo and Gray. In LBG algorithm quantizer is designed for memoryless Gaussian sources with a mean-squared error distortion. Block quantizers with arate of one bit per symbol and with block lengths of 1 through 6 have been designed by Linde , Buzo and Gray. They compressed a traditional 6000 bits /s Linear Predictive Coded (LPC) speech to a rate of 1400 bits/s.

Lloyd's treatment is found in (12).For design based on empirical data; it was first used for VQ in 1977 by Chen for design of 2-dimensional quantizers

(13) and by Hilbert (14) for the design of multispectral image compression system.

The earliest use of randomness for VQ design was proposed in (10), where noise is added to the training vectors prior to Lloyd iteration and the variance of noise is gradually decreased to zero. A convenient and effective Stochastic Relaxation (SR) algorithm for codebook design was introduced in (16) & is a simpler version of earlier methods proposed in (15) and (17). It would be of great value if we find a codebook design algorithm that leads to a global optimum without the typically very time consuming process that is introduced by SA with effective cooling schedules. The work of Rose et. al.(18)(19) offers a novel approach to nonconvex optimization called deterministic annealing (DA),that appears to capture the benefits of Stochastic Annealing (SA) for VQ codebook design without any randomness in the design process. It is conceptually similar to the technique of fuzzy clustering described in (20), (21).

A standard method for designing the tree structure (consisting of set of test vectors for each node and associated codebook) is based upon application of GLA to successive stages using a training set. This procedure was proposed in (10), is a variation of standard splitting method of designing full search VQ. Another approach to designing a tree-structured VQ follows from viewing it as a classification tree. Briman,Friedman,Olshen & Stone (22) provide a variety of algorithms for “growing” such trees. Similar technique is proposed by Makhoul , Roucos and Gish (23). Yet another strategy for choosing a node to split is to perform a principal components analysis and split in the direction of maximum variance (24).

A specific design technique for bit allocation technique for bit allocation in classified VQ is considered in Riskin (25). In image coding it has been found that certain classes are of greater perceptual significance than others so that it can be advantageous to make sub-codebook size for such a class disproportionately large compared to relative occurrence of such vectors in the training set (26).

Image transform VQ using DCT is treated in Aizawa et. al. (27). Early work on VQ in transform domain is found in King and Nasarbadi (28). A similar application for speech compression can be found in Adlersberg and Cuperman (29).Developments of VQ sub-band coding for speech is found in (30) and (31) and for images in (32).

It has been claimed that wavelet decompositions offer better models of human auditory/ visual processing. Antonini,Barlaud,Mathieeu and Daubechies (33) reported good quality images using vector quantizer based wavelet transform coding combined with entropy coding at 1 bpp.

Shape-gain VQ was introduced in (35) and optimized in (34). It is particularly useful in speech and audio coding where the waveform has a wide dynamic range of short term power levels. Shape-gain VQ designed for LPC-VQ with the residual energy playing the role of gain is found in (36).Multistage / cascaded VQ is valuable in number of speech and image coding applications. Sometimes it is referred to as residual VQ (37) (38).

Constrained Storage VQ (CSVQ) has been applied successfully to wideband audio coding where tree structured code-books previously solved problem of encoding complexity and CSVQ solved remaining problem of astronomic storage complexity (39). Hierarchical VQ (HVQ) was introduced in (40). Other variations of HVQ were reported in (41), (42), (43) and (44).

Selection of nearest neighbors in lattice to a given input vector is considered in depth in (45)(46)(47)(48)(50).The indexing problem is treated in (48)(49).Most widely studied fast search technique is based on the use of k-d trees (51),(52)(53)(54)(7).A k-d tree is a binary tree with a hyper plane decision test at each node where each hyper plane is orthogonal to one of the coordinate axes of k- dimensional space and a set of buckets or terminal nodes.

The technique of predictive Vector Quantization (PVQ) was introduced in 1982. More extensive studies of PVQ followed in (55) and (56).Both finite- state and predictive vector quantizer (FSVQ , PVQ) can be considered as special cases of more general class of VQ systems with memory called recursive vector quantization ./ Feedback vector quantization (57). Recursive vector quantization in generalization to vector of scalar recursive quantizers are considered in (58)(59)(60).

The Lloyd-style trellis code improvement algorithm was developed by Stewart .Mean-adaptive VQ used for image coding is given by Baker and Gray. Performance of shape-gain VQ is enhanced by adapting gain codebook while maintaining fixed shape codebook with unit norm code vectors.

Vector Excitation Coding (VXC) and in a more specific context Code Excited Linear Prediction (CELP) is a powerful and widely used signal coding technique. These LPC-VQ (Linear Predictive Coding) systems encode the speech at low rates (61) (62) (63) (64).Levinson-Durbin algorithm followed by quantization of reflection coefficients yields Residual Excited Linear Predictive (RELP) system (65) (66).A large family of variable rate coding methods exists.

No technique has been applied for the design of the code book with the pdf in consideration. This is the prime motivation to carry out the work in this area.

1.2 Major Contributions of the Thesis:

Major contributions of this thesis are:

- Comprehensive study of Concept of Vector Quantization.
- Review of Waveform Coding.
- Design and performance of Unconstrained Vector Quantizers along with codebook designed with the pdf in consideration.
- Design and performance of constrained Vector Quantization Techniques along with codebook designed with the pdf in consideration.
- Comparison of various techniques which are done by authors.

1.3 Organization of the Thesis:

The thesis is organized in total ten chapters as described below:

Chapter:1	It provides details about motivation for research work, major contributions of the thesis and organization of the thesis.
Chapter: 2	It explores comprehensive study of concept of vector quantization.
Chapter: 3	It reviews waveform coding techniques.
Chapter: 4	It describes the design and performance of unconstrained vector quantizers for full search vector quantizer.
Chapter: 5	It describes the design and performance of constrained vector quantization techniques for tree structured vector quantizer.
Chapter: 6	It describes the design and performance of constrained vector quantization techniques for multi stage vector quantizer.
Chapter: 7	It describes the design and performance of constrained vector quantization techniques for trellis coded vector quantizer.
Chapter: 8	This compares the performance of various techniques proposed in previous chapters.
Chapter: 9	It contains discussion of the results and conclusions as well as scope of future work.
Chapter: 10	It contains bibliography.

Chapter 2

Vector Quantization

2.1 Basics of Vector Quantization

Vector quantization is a lossy data compression technique. To study in detail it is essential to learn basics of vector quantization (VQ).

A vector quantizer Q of dimension k and size N is a mapping form a vector (or a “point”) in k-dimension Euclidean space, R^k, into a finite set C containing N output or reproduction points called *code vectors* or *code words.* Thus,

$$Q : R^k \to c, \tag{2.1}$$

Where, $c = (y_1, y_2, ..., y_N)$ and, $y_i \in R^k$ for each, $i \in J \equiv \{1, 2, - - -, N\}$. The set *c* is called the codebook or the code and has size N, meaning it has N distinct elements, each a vector in R^k. The resolution, code rate, or, simply, rate of a vector quantizer is $r = (\log_2 N)/k$, which measures the number of bits per vector component used to represent the input vector and gives an indication of the accuracy or precision that is achievable with a vector quantizer if the codebook is well-designed. If the codebook represents Euclidean space partitions which is very similar to the distribution of the input samples then the codebook is well designed. Typical value of N can be taken 16 or higher. It is important to recognize that for a fixed dimension k the resolution is determined by the size N of the codebook and not by the number of bits used to numerically specify the code vectors stored in the codebook.

The codebook is typically implemented as a table in a digital memory and the number of bits of precision used to represent each component of each code vector does not affect the resolution or the bit-rate of the vector quantizer; it is of concern only in connection with storage space limitations and with the question of adequate precision in describing a well-designed codebook.

- Associated with every N point vector quantizer is a partition of R^k into N regions or cells, R_i for $i \in J$. The ith cell is defined by

  $$R_i = \{x \in R^k : Q(x) = y_i\} \quad (2.2)$$

- Sometimes called the inverse image or pre-image of Yi under the mapping Q and denoted more concisely by $R_i = Q^{-1}(y_i)$.

  From the definition of the cells, it follows that

  $$\bigcup_i R_i = R^k and\ R_i \cap R_j = \phi\ for\, i \neq j \quad (2.3)$$

  So that the cells from a partition of R^k .This represents condition to be satisfied for the inverse mapping of y_i.

- A cell that is unbounded is called an over loaded cell. Abounded cell i.e., one having finite (k-dimensional) volume, is called a granular cell. The collection of all granular cells is called the granular region.

An important property of a set in R^k is *convexity*. Recall that in two or three dimensions, as set is said to be convex if given any two points in the set, the straight line joining these two points is also a member of the set. This familiar idea remains applicable in R^k. A set $S \in R^k$ is convex if a and $b \in s$ implies that

$$\alpha a + (1-\alpha)\, b \in s \text{ for all } 0 < \alpha < 1 \quad (2.4)$$

2.2 Definitions

A vector quantizer is called *regular* if,

1. Each cell, R_i, is a convex set, and
2. For each i, $y_i \in R_i$.

It is also convenient to define a *polytopal vector quantizer* as regular quantizer whose partition cells are bounded by segments of hyper plane surfaces in k dimensions. Equivalently, each partition region is a regular polytope and consists of an intersection of a finite number of half spaces of the form $\{x \in R^k: u_v.\, x + \beta_v \geq 0\}$.

For a thorough treatment of polytopes, the generalization of polyhedra. The faces of a polytopal cell are hyper plane segments of dimension less than k that bound the cell, so that every point on one side of the face is inside the cell and every point on the other side of the face is outside the cell. Usually, a face refers to a k - 1dimensional hyper plane segment for a cell in k dimensions. Note that the definition of a regular vector quantizer is consistent with the scalar case and that a regular quantizer in the one-dimensional case is always polytopal.

- A vector quantizer is said to be *bounded*, if it is defined over a bounded domain, $B \subset R^k$, so that every input vector, x, lies in this set. The volume of the set B, denoted by V (B) and given by

$$V(B) = \int_B x \, dx, \tag{2.5}$$

is therefore finite. A bounded quantizer does not have any overload regions in its partition. Overload region does not have limit on sample values.

- A vector quantizer can be decomposed into two component operations, the vector encoder and the vector decoder. The encoder ε 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,

  $\varepsilon: R^k \rightarrow J$ and $D : J \rightarrow R^k$ (2.6)

It is important to note that 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 task of the encoder is either implicitly or explicitly to identify in which of N geometrically specified regions of k space the input vector lies. Contrary to popular belief, ideally the encoder does not fundamentally need to know the codebook to perform its function. Because encoder has to identify the partition to which the input vector belongs. On the other hand, the decoder is simply a table lookup that is simply and fully determined by specifying the codebook. This is the prime reason why VQ is very popular where decoding is less intensive.

The decoder does not need to know the geometry of the partition to perform its job. Later we shall see that for most vector quantizers of practical interest, the codebook provides sufficient information to characterize the partition and in this case, using the codebook as the data set, which implicitly specifies the partition, can perform the encoder operation.

The overall operation of VQ can be regarded as the cascade or composition of two operations:

$$Q(x)=D(\varepsilon(X)) \tag{2.7}$$

Occasionally it is convenient to regard a quantizer as generating both an index i, and a quantized output value, Q (x). The decoder is sometimes referred to as an "*inverse quantizer*".

Figure 2.1 illustrates how the cascade of an encoder and decoder defines a quantizer.

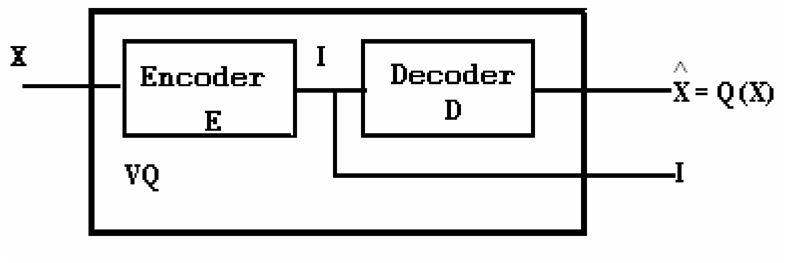


Figure: - 2.1 A Vector Quantizer as the Cascade of an Encoder and Decoder.

In the context of a digital 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. If a sequence of input vectors is to be quantized and transmitted, then the bit-rate or transmission rate R, in bits per vector, is given by R = k r, where r is the resolution and k the vector dimension. If we let f_v denote the vector rate, or the number of input vectors to be encoded per second, then the bit-rate, R_s in bits per second is given by

$$R_s = k\, r\, f_v \tag{2.8}$$

Of particular interest is the case of a scalar waveform communication system where each vector represents a block of contiguous samples of the waveform. The vector sequence, called a blocked scalar process, then corresponds to consecutive blocks of the waveform and the vector dimension, k, and the sampling rate, f_s, measured in samples per second, together determine the vector rate, in this case $f_v = f_s / k$ vectors per second. The bit-rate in bits per second is therefore, $R_s = r\ f_s$ which is independent of the dimension k. We distinguish between resolution and rate since there are important applications of VQ where the vectors are extracted parameters of a signal rather than blocked samples of a waveform.

Vector Quantization is not merely a generalization of scalar quantization. In fact, it is the "ultimate" solution to the quantization of a signal vector. Because VQ represents the signal with much less number of bits compared to scalar quantization. No other quantization technique exists that can do better than VQ.

2.3 Examples of Vector Quantization

It is convenient to view the operation of a vector quantizer geometrically, using our intuition for the case of two- or three-dimensional space. Thus, a 2-D quantizer assigns any input point the plane to one of a particular set of N points or locations in the plane. As a simple illustration consider a map of a city that is divided into school districts and the codebook is simply the location of each school on the map. The “input” is the location of a particular child’s residence and the quantizer is simply the rule that assigns each child to a school according to the child’s location.

Figure 2.2 depicts an example of a two-dimensional (2-D) quantizer that is neither polytopal nor regular since the cells have faces (one-dimensional boundaries) that are not segments of hyper planes (straight line segments) and the cells are not convex. The dots represent code vectors in a 2 dimensional space (the Plane) and the region containing each code vector is a partition cell.

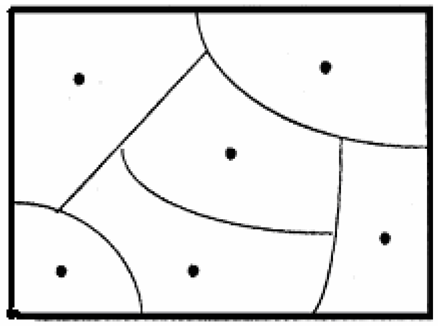


Figure: - 2.2 A Non Regular Quantizer

Figure 2.3 shows a two dimensional regular quantizer. Whose bounded cells are polygons (closed polytopes in two dimensions).

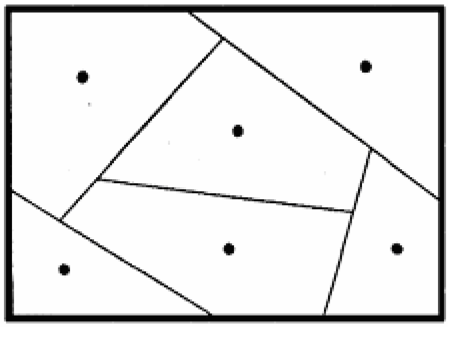


Figure: - 2.3 A Regular Quantizer

On the generality of VQ, this case can be considered as a degenerate special case of vector quantization of the vector $x = (x_1,x_2)$ where the vector quantizer is given by,

$$Q(X) = (Q1(XI), Q2(X2)) \tag{2.9}$$

where Q_l and Q_2 are the scalar quantizers for x_1 and x_2, respectively.

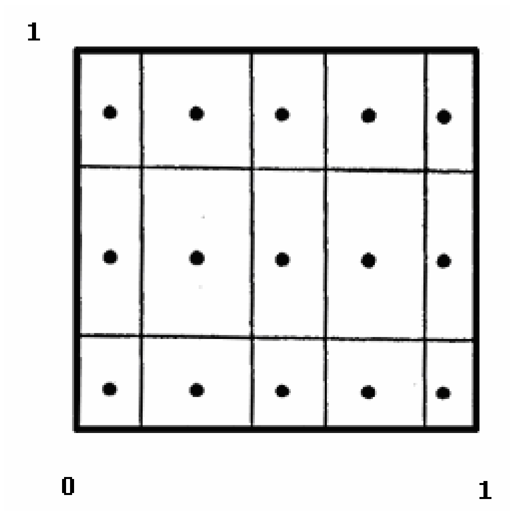


Figure: - 2.4 VQ Based Upon Scalar Quantization

Figure 2.4 shows the resulting vector quantizer corresponding to a particular choice of scalar quantization for each variable. Separate quantization for each variable is not preferable as it does not take advantage of the correlation between the variables. We note in passing that this is an example of a product VQ because the overall VQ is formed as a Cartesian product of smaller dimensional VQs.

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 as indicated in the examples of Figures 2.2 and 2.3. 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. (A polyhedron is a polytope in three dimensions.) Extending this idea to k dimensions, it is clear that scalar quantization of the components of a vector always generates a restricted class of vector quantizers where the faces are (k-1)-dimensional hyper planes 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. VQ offers structural freedom by having partitions of any arbitrary shape unlike partitions with rectangular box-like shapes in scalar quantization.

2.4 Vector Quantizer Performance Measurement

A distortion measure d is an assignment of a nonnegative cost d (x, x) associated with quantizing any input vector x with a reproduction vector x. Given such a measure we can quantify the performance of a system by an average distortion $D = Ed(x, \hat{x})$ between the input and the final reproduction. Generally the performance of a compression system will be good, if the average distortion is small. In practice, the overall measure of performance is the long-term sample average or time average

$$\hat{d} = \lim_{n\to\infty} 1/n \sum_{i=1}^{n} d(X_i, \hat{X}_i) \tag{2.10}$$

Where $\{x_i\}$ is a sequence of vectors to be encoded. If the vector process is stationary and ergodic, then with probability one the above limit exists and the statistical expectation, i.e., $\hat{d} = D$. Stationarity and ergodicity, however are not, necessary and similar properties can hold under more general conditions.

Let X denotes a continuously distributed random vector in R^k with a specified pdf (In this case the pdf refers to a joint probability density function of the vector components, x_i for i=1, 2 ... k. Then the average distortion can be expressed as

$$D = Ed(X, Y) = \int_{R^k} d(x, Q(x)) f_X(x)dx \tag{2.11}$$

and using the partition and codebook for the given quantizer Q, we get

$$D = \sum_{j=1}^{N} \int_{R_j} d(x, y_j) f_x x \; dx \tag{2.12}$$

$$D = \int \sum_{j=1}^{N} p_j d(x, \; y_j) f_x(x)dx \tag{2.13}$$

$$= \sum_{j=1}^{N} p_j \, E\left[d(x, \; y_j) \; \middle| X \in R_j\right] \tag{2.14}$$

where $p_j = P(X \in R_j)$ and $f_{x/j}(x/j)$ is the conditional probability density of x given that $x \in R_j$.

Occasionally, it is preferable to use a worst-case distortion as a measure of performance rather than an average value. With respect to In arbitrary measure d (x, y) of distortion, this is simply defined as the maximum attainable distortion for a given quantizer:

$$D_{max} = \max_{x \in B} d(x, Q(x)) \tag{2.15}$$

Where B is the closed subset of R^k to which the input X is confined that is, B is the domain over which the pdf of X is nonzero. X outside this region B has zero probability so it does not affect $D_{max.}$ In some cases, this maximum may not exist since the distortion can become arbitrarily large and this measure is then not meaningful. Use of the maximum distortion as a performance measure is limited to the case of bounded random vectors or quantizers with countably finite codebook sizes with no overload regions. For the most part, we shall focus on the statistical average of the distortion as a performance measure.

Ideally, a distortion measure should be tractable to permit for guiding the actual encoding process for encoders, which select a nearest neighbor or minimum distortion output (we shall see that as with scalar quantizers, this form of encoder is optimal for a given codebook). It should also be subjectively meaningful so that large or small average distortion values correlate with bad and good subjective quality as perceived by the ultimate user of the reproduced vector sequence. This subjective measure is more useful as distortions with different objective values are difficult to differentiate by user.

The most convenient and widely used measure of distortion between an input vector x and a quantized vector Y = Q (x), is the squared error or squared Euclidean distance between two vectors defined as

$$d(x,\hat{x}) = \|x - \hat{x}\|^2 \qquad (2.16)$$

$$\equiv (x - \hat{x})^t (x - \hat{x}) \qquad (2.17)$$

$$= \sum_{i=1}^{k} (x_i - x_{\hat{i}})^2 \qquad (2.18)$$

Euclidean distance can also be modeled with random process that has continuous distribution. If the input and reproduction vectors are complex, then this becomes

$$\begin{aligned} d(x,\hat{x}) &= \|x-\hat{x}\|^2 \\ &\equiv (x-\hat{x})^*(x-\hat{x}) \\ &= \sum_{i=1}^{k} |x_i - \hat{x}_i|^2 \qquad (2.19) \end{aligned}$$

The average squared error distortion or, more briefly, the *average distortion* (when no confusion with other distortion measures arises) is defined as

$$D = Ed(x,\hat{x}) = E(\|x-\hat{x}\|^2) \qquad (2.20)$$

This measure is frequently associated with the energy or power of an error signal and therefore has some intuitive appeal in addition to being an analytically tractable measure for many purposes.

Alternative distortion measures may also be defined for *assessing dissimilarity* between the input and reproduction vectors. Many of the measures of interest for VQ have the form

$$d(x,\hat{x}) = \sum_{i=1}^{k} d_m(x_{i,}\hat{x}_i) \qquad (2.21)$$

Where d_m $(x,\hat{x})$ is a scalar distortion measure (often called the per-letter distortion) as in one-dimensional quantization.

Any distortion measure having this additivity property with the same scalar distortion measure used for each component is called an additive or single letter distortion measure and is particularly appropriate for waveform coding where each vector component has the same physical meaning, being a sample of a waveform. Of particular interest is the case where the scalar distortion measure is given by $d_m(x, \hat{x}) = |x - \hat{x}|$ for positive integer values of m When m =1, this specializes to the l_1 norm of the error vector, $|x - \hat{x}|$ When m = 2, we obtain the squared error measure lm already discussed. The mth root of d_m is the lm norm of the error vector $|x - \hat{x}|$.

Another distortion measure of particular interest is the *weighted squared error* measure

$$d(x, y) = (x - y)^t W(x - y) \quad (2.22)$$

Where W is a symmetric and positive definite weighting matrix and the vectors x and y are treated as column vectors. Note that this measure includes the usual squared error distortion in the special case where W = I, the identity matrix. In the case where W is a diagonal matrix with diagonal values $w_{ii} > 0$, we have

$$d(x, y) = \sum_{i=1}^{k} w_{ii}(x_i - y_i)^2 \quad (2.23)$$

which is a simple but useful modification of the squared error distortion that allows a different emphasis to be given to different vector components.

All of the distortion measures discussed so far are symmetric in their arguments x and y. It is sometimes convenient and effective to choose a weighting matrix W(x) (assumed to be symmetric and positive definite for all x) that depends explicitly on the input vector x to be quantized in order to obtain perceptually motivated distortion measures for both speech and image compression. In this case, the distortion

$$\mathrm{d(x, y) = (x - y)^t W(x)(X - Y)} \tag{2.24}$$

is in general asymmetric in x and y. As an example of such a distortion measure, let W(x) be $\|X\|^{-2}$ I, where I is the Identity matrix. Here, the distortion between two vectors is the noise energy to signal energy ratio:

$$d(x, \hat{x}) = \frac{\|x - \hat{x}\|^2}{\|x\|^2} \tag{2.25}$$

This allows one to weight the distortion as being more important when the signal is small than when it is large. Note the distortion measure is not well defined unless $\|x\|$ > 0.

Finally, we define *the maximum or l_∞ norm distortion* measure by:

$$d_{\max}(x, \hat{x}) = \max_i \left| x_i - \hat{x}_i \right| \tag{2.26}$$

Where, the distortion is determined by the component of the error vector $x - \hat{x}$ that contributes the largest absolute error. It is a well-known mathematical result that the l_m norm approaches the l_∞ norm as m $\rightarrow \infty$ Thus,

$$\lim_{m \to \infty} [\mathrm{d_m}\ (x, \hat{x})]^{1/m} = \mathrm{d_{max}}\ (x, \hat{x}) \tag{2.27}$$

2.5 Summary

This chapter described the definition, concept and basic types of VQ. It also includes primary structure of vector quantizer As a performance measure criteria distortion calculation is explained.

Chapter 3

Waveform Coding techniques

Waveform codecs have been comprehensively characterized by Jayant and Noll(67).In general, waveform codecs are designed to be signal independent. They are designed to map the input waveform of the encoder into a facsimile-like replica of it at the output of the decoder. Because of this advantage, they can also encode a secondary type of information such as signaling tones, data, or even music. Because of this signal transparency their coding efficiency usually quite modest. The coding efficiency can be improved by exploiting some statistical signal properties, if the codec parameters are optimized for the most likely categories of input signals, while still maintaining good quality for other types of signals as well. The waveform codecs are further subdivided into time-domain waveform codecs and frequency-domain waveform codecs.

3.1 Time-domain Waveform Coding

The most well-known representative of signal-independent time-domain waveform coding is the A-law companded pulse code modulation (PCM) scheme. This coding has been standardized by the CCITT at 64 kbits/s, using non-linear companding characteristics to result in near-constant signal-to-noise ratio (SNR) over the total input dynamic range. Upon quantizing this companded signal, large-input samples will tolerate higher quantization noise than small samples.

Also well-known is the 32 kbits/s adaptive differential PCM (ADPCM) scheme standardized in the ITU Recommendation G.721 and the adaptive delta modulation (ADM) arrangement, where usually the most recent signal sample or a linear combination of the last few samples is used to form an estimate of the current one. Then their difference signal, the prediction residual, is computed and encoded with a reduced number of bits, since it has a lower variance than the incoming signal. This estimation process is actually linear prediction with fixed coefficients. However, owing to the nonstationary statistics of speech/image, a fixed predictor cannot consistently characterize the changing spectral envelope of speech signals.

All in all, time-domain waveform codecs treat the signal to be encoded as a full-band signal and attempt to map it into as close a replica of the input as possible. The difference among various coding schemes is in their degree and way of using prediction to reduce the variance of the signal to be encoded, so as to reduce the number of bits necessary to represent it.

3.2 Frequency Domain Waveform Coding

In frequency-domain waveform codec, the input signal undergoes a more or less accurate short-time spectral analysis. The signal is split into a number of sub-bands, and the individual sub-band signals are then encoded by using different number of bits in order to obey rate-distortion theory on the basis of their prominence. The various methods differ in their accuracies of spectral resolution and in the bit-allocation principle (fixed, adaptive, semi-adaptive). Two well-known representatives of this class are sub-band coding (SBC) and adaptive transform coding (ATC).

3.3 Waveform Coding Techniques based on Vector Quantization

There are number of waveform coding techniques using vector quantization. Initially Nearest Neighbor Quantizers were used but they were unconstrained in the sense that they require large memory for good performance. There is a variety of constrained coding schemes that provide reduced complexity and better performance in trade for a tolerable loss of optimality. Included are tree-structured vector quantization (TSVQ),classified vector quantizers, transform vector quantizers, product codes such as gain/shape and mean-residual vector quantizers , and multistage vector quantizers.There exists fast search algorithms for codebook searching , nonlinear interpolative coding , and hierarchical coding.

Vector quantizers with memory are called recursive quantizers or feedback vector quantizers.In predictive vector quantization (PVQ) vector predictors are used to form a prediction residual of the original input vector and resulting residual is quantized. In finite-state vector quantization (FSVQ) encoder and decoder are finite-state machines. Like a predictive VQ, a finite-state VQ uses the past to implicitly predict the future and use a codebook matched to the likely behavior. Unlike a predictive VQ, a finite-state VQ is limited to only a finite number of possible codebooks.

Tree and trellis encoding systems have decoders like those of predictive and finite-state vector quantizers, but the encoders are allowed to “look ahead” into the future before making their decisions as to which bits to send. At the cost of additional delay, such coding methods can provide improved performance by effectively increasing the input vector size while keeping complexity manageable.

In adaptive vector quantizers codebooks are allowed to change in a slow manner relative to the incoming data rate so as to better track local statistics variations. Both forward and backward adaptation is possible. More complicated adaptive coding schemes such as residual excited linear prediction (RELP) and code excited linear prediction (CELP) VQ are used in mobile communication.

Vector quantizers with variable-rate coding use more bits for active signals and fewer for less active signals while preserving an overall average bit rate. Such coding systems can provide a significantly better tradeoff between bit rate and average distortion, but they can be more complex and can require buffering if they are used in conjunction with fixed rate communication links. The performance improvement often merits any such increase in complexity, and the complexity may in fact be reduced in applications that are inherently variable rate such as storage channels and communication networks. In statistical pattern classification coder design algorithms are developed for unbalanced tree-structured vector quantizers which are very similar to variable-rate vector quantizers.

3.4 Waveform properties and Encoding Delay

Coder complexity is a function of the signal processing involved. It is also related to encoder delay or encoder memory, which shows the extent to which a waveform has to be observed in order for the coder to exploit waveform structure for economical digital representation. Examples are listed in Table 3.1.

Table 3.1 A listing of coding systems in terms of encoding delay

Waveform Memory Used in Coding	**Coding System Example**
Zero Memory	Instantaneous Quantizers: PCM
Short-term Memory	Adaptive Quantizers
	Differential PCM systems with near-sample-based predictions
	Delta modulators
	Sub-Band Coders for Speech
	Vector Quantizers : Tree- and Trellis Coders
	Intraframe Transform Coders for Images
Long-term Memory	Differential PCM systems with distant-sample-based predictions
	Transform Coders for Speech
	Intraframe Transform Coders for Images

This list attempts to grade quantizer and coder classes in order of increasing encoder memory or delay. The counter-examples are also possible. For example, in speech coding, depending on the extent of memory utilized, sub-band coders, vector quantizers and tree coders may well be included in the class of coders with long-term memory.

The frame period in video is either 33.33 ms or 40 ms, depending on power line frequency, while the pitch period in speech can be in the range 3 to 15 ms, depending on speaker. The use of long-term memory in Table 3.1 indicates waveform observations over duration in order of the pitch period in speech, and the frame period in video. This implies transmission delays that can be several tens of milliseconds in both speech and image coding. Such delays can have important in aspect of waveform communications such as echo control in long-distance two-way transmissions.

3.5 Selection of Vector Quantization Techniques

There are many techniques for waveform coding using Vector Quantization. Out of this, Tree Structured Vector Quantization, Multi Stage Quantization and Trellis Coded Vector Quantization have been selected randomly. A brief comparison along with advantages and disadvantages of these VQ techniques are discussed below.

In m-ary tree search with balanced tree , the input vector is compared with m predesigned test vectors at each stage or node of the tree.The tree structured VQ has the advantage that an m-ary tree with d stages has search complexity proportional to md rather than m^d. The disadvantage of tree structured VQ is that storage requirements are increased compared to Full search VQ. Another disadvantage of tree structured VQ is that there is additional delay compared to Full search VQ.

In Multi Stage Vector Quantization (MSVQ) each stage simply quantizes residual of the previous stage. Multi Stage VQ does not require high codebook storage requirements as Tree Structured VQ. MSVQ does not require huge training set as TSVQ. Complexity and storage requirements are greatly reduced by using Multi Stage VQ.MSVQ has the disadvantage that every stage will add quantization error and hence increase quantization noise.

The trellis consists of the set of all paths starting at the root node and traversing one of the K state nodes at each time instant and terminating at a fixed node at time L-1. L is the encoder delay or search depth. We do not have to search a block code of 2^L by computing all 2^L possible distortions. Instead we compute a sequence of K distortions. The complexity grows with the number of states, not with L. The disadvantage of trellis coded VQ is that encoding delay can be intolerable in some applications. Another disadvantage of trellis coded VQ is that encoding complexity grows exponentially and can quickly become prohibitive for high or even moderate rates.

3.6 Summary

The chapter describes broad categories of waveform coding techniques namely time domain, frequency domain and waveform coding techniques based on vector quantization. Examples are given for coders with different encoder delay depending upon waveform properties. A brief comparison along with advantages and disadvantages of Vector quantization techniques selected are also discussed.

Chapter 4

Full Search Vector Quantizer

4.1 Full Search Vector Quantizer

A full search vector quantizer considers distortion between the sample and every codeword of the codebook. It will then try to minimize the distortion function. A full search vector quantizer is normally called Nearest neighbor vector quantizer. An important special class of vector quantizers that is of particular interest, called *Voronoi or nearest neighbor* vector quantizers has the feature that the partition is completely determined by the codebook and a distortion measure. The term “vector quantizer” is commonly assumed to be synonymous with “nearest neighbor vector quantizer”. All advantage of such all encoder is that the encoding process does not require any explicit storage of the geometrical description of the cells. Instead a conceptually simple algorithm can encode by referring to the stored codebook.

4.2 Nearest Neighbor Vector Quantizer

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

$$d(x, y) = \| x - y \|^2 = \sum_{i=1}^{k} (x_i - y_i)^2 \tag{4.1}$$

This is, the simplest measure and the most common for waveform coding. While not subjectively meaningful in many cases, generalizations permitting input dependent weightings have proved useful and often only slightly more complicated.

We define a *Voronoi or nearest neighbor (NN)* vector quantizer as one whose partition cells are given by

$$R_i = \{x : d(x, y_i) \le d(x, y_j) \ \ all \ \ j \in J\} \tag{4.2}$$

NN VQ will constitute a regular quantizer if centroid is a representative of regular sized cells. NN VQ will constitute a irregular quantizer if centroid is a representative of irregular sized cells. In order for the Cells to constitute a partition, each boundary point, must be uniquely assigned to one cell. This modification of the above definition is readily handled by assigning x to be a member of R_m where m is the smallest index i for which $d(x, y_i)$ attain its minimum value.

The following simple algorithm gives the most direct encoding algorithm for an NN encoder.

Nearest Neighbor Encoding Rule
1. Set $d=d_o$,j= 1, and i=1.
2. Compute $D_j= d(\mathbf{x}, \mathbf{y_j})$.
3. If $D_j < d$, set $D_j \rightarrow d$. Set $j \rightarrow i$.
4. If $j < N$, set $j + 1 \rightarrow j$ and go to step 2.
5. Stop. Result is index i.

Table 4.1 Nearest Neighbor Encoding Rule

The resulting value of i gives the encoder output C(x) and the final value of d is the distortion between x and y_i. The initial value, do, must be larger than any expected distortion value and is usually set to the largest positive number that can be represented with the processor's arithmetic.

The key feature of the above encoding algorithm is that no geometrical description of the partition is needed to perform the encoding rule. Thus the encoder operation may be described by x

$$\varepsilon(x) = c(x, C) \tag{4.3}$$

where the functional operation described by c(., .) is independent of the specific quantizer and depends only on the distortion measure. This important concept is central to the implementation of virtually all vector quantizers in use today. Figure 4.1 illustrates the form of the NN encoder where the Codebook C is a read-only memory that is accessed by the NN encoder box that implements the function c(., .).

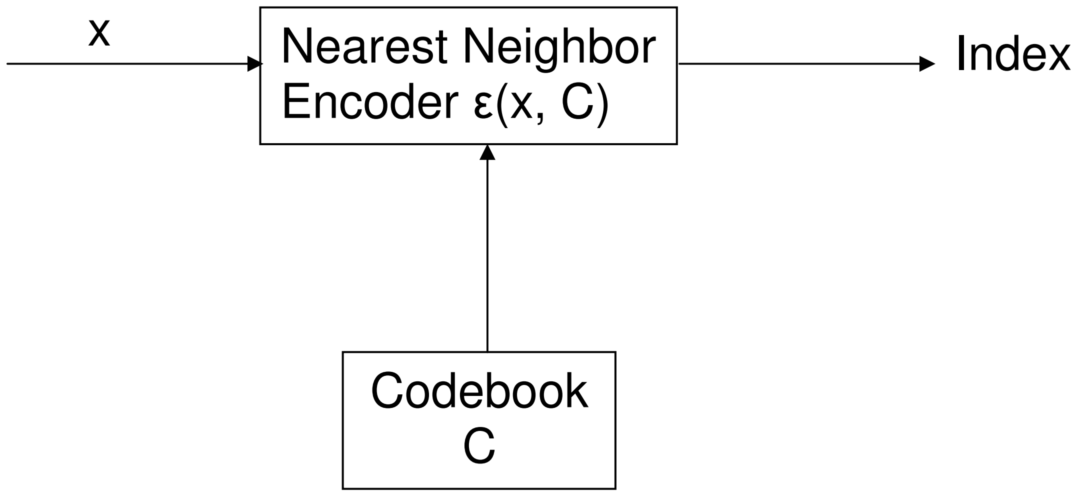


Figure : - 4.1 - Nearest Neighbor Encoder with a Codebook ROM

Virtually any computable distortion measure can be used for a NN encoder. Later, however, we shall see that other properties of a distortion measure are required to yield a statistically based performance measure that is amenable to optimal codebook design. In general the NN cells defined above need not be polytopal or even convex.

We defined that the NN cells can be expressed as:

$$\mathrm{R_i} = \bigcap_{\substack{j=1 \\ j \neq i}}^{N} H_{ij} \tag{4.4}$$

Where,

$$\mathrm{H_{ij}} = \{\mathrm{x} : \|\mathrm{x} - \mathrm{y_i}\|^2 \leq \|\mathrm{x} - \mathrm{y_j}\|^2\} \tag{4.5}$$

Expanding the squared norms and simplifying, we get

$$\mathrm{H_{ij}} = \{\mathrm{x} : \mathrm{u_{ij}} .\mathrm{x} + \beta_{ij} \geq 0\} \tag{4.6}$$

Where

$$\mathrm{u_{ij}} = 2(\mathrm{y_i} - \mathrm{y_j}) \tag{4.7}$$

and,

$$\beta_{ij} = \|y_j\|^2 - \|y_i\|^2 \tag{4.8}$$

Thus, we see that the NN cells of a Voronoi quantizer are formed by the intersection of half-spaces so that every Voronoi quantizer is polytopal and its half-spaces are explicitly determined from the code vectors y_i of its codebook. This convenient property is a direct consequence of the fact that the squared error distortion measure is a quadratic function of the code vector components. Other more complex distortion measures which depend on higher order powers of the input vector components will not give rise to polytopal NN quantizers.

For the squared error distortion measure or squared Euclidean distance, the partition cells of the NN encoder not only are polytopal but have all explicitly determined specification derived from the code vectors. This allows a simplification of the general tertiary structure.

Finally, we point out a simple and important variation of the nearest neighbor-encoding algorithm described earlier. It follows either from the preceding discussion or directly from the squared Euclidean distance measure that the nearest neighbor search can be performed by evaluating scalar products rather than taking squared norms since

$$\min_i \|x - y_i\|^2 = \max_i{}^{-1}[x^t y_i + \alpha_i] \tag{4.9}$$

Where,

$$\alpha_i = -\|y_i\|^2 \big/ 2 \tag{4.10}$$

and the values of α_i can be precomputed and stored along with the codebook, we have the following algorithm

Alternate Nearest Neighbor Encoding Rule
1. Set $f = f_o$, $j = 1$, and $i = 1$.
2. Compute $F_j = \mathbf{x}^{\mathbf{t}}\ \mathbf{y}_{\mathbf{j}} + \alpha_j$
3. If $F_j > f$, set $F_j \rightarrow f$. Set $j \rightarrow i$.
4. If $j < N$, set $j + 1 \rightarrow j$ and go to step 2.
5. Stop. Result is index i.

Table 4.2 Alternate Nearest Neighbor Encoding Rule

The resulting value of i gives the encoder output C(x) and the final value of determines the distortion between x and y_i according to

$$d(x, y_i) = \|x\|^2 - 2f \tag{4.11}$$

The initial value, f_0, must be smaller than any expected distortion value and is usually set to zero. The disadvantage of this alternate approach is that values of α_i needs to be stored along with the code vectors. Another disadvantage of this approach is that for computing parameter F multiplication and addition is required. So storage and computing requirements are more with this approach.

In real-time implementations of VQ encoders, the preferred choice of search algorithm depends on the particular processor architecture. The earliest reported hardware implementation of VQ implemented the nearest neighbor search algorithm and used off-the-shelf components and the Z80A, an 8-bit microprocessor. Most future applications of VQ for speech compression algorithms will be ASIC implementation on single-chip programmable signal processors.

4.3 Optimality Conditions for VQ

The principal goal in design of vector quantizers is to find a codebook, specifying the decoder, and a partition or encoding rule, specifying the encoder that will maximize an overall measure of performance considering the entire sequence of vectors to be encoded over the lifetime of the quantizer. The overall performance can be assessed by either a statistical average of a suitable distortion measure or by a worst-case value of distortion. We focus here only on statistical criteria. The statistical average of the distortion for a vector quantizer Q (.), can be expressed as

$$D = Ed(X, Q(X)) = \int d(x, Q(x)) f_x(x)\, dx \tag{4.12}$$

Where $f_x(x)$ is the (joint) pdf of the vector X and the integration above is understood to be a multiple integral over the k-dimensional space. When the input vector has a discrete distribution, a case that will also be of interest to us later, it is more convenient to avoid the use of a pdf made up of delta functions and instead describe the distribution by the probability mass function, (pmf) denoted by px(x). Then we have

$$D = Ed(X, Q\,(X)) = \sum_i d(x_i, Q(x_i)) px(x_i) \tag{4.13}$$

where $\{x_i\}$ are the values of x that have nonzero probability. It is a common practice in engineering to use the Dirac delta function in order to retain the use of the pdf for a discrete probability distribution. Here one adds the probability masses of points to find the overall probability.So for simplifying the computational requirements continuous distributions are used for discrete process. Distribution is normally assumed apriori.

We assume that the codebook size N is given, the k-dimensional input random vector, x, is statistically specified, and a particular distortion measure d(.,.) has been selected. We wish to determine the necessary conditions for a quantizer to be optimal in the sense that it minimizes the average distortion for the given conditions. As in the scalar case, we proceed by finding the necessary condition for the encoder to be optimal for a given decoder. Then for a given encoder we find the necessary condition for the decoder to be optimal. Recall that the encoder is completely specified by the partition of R^k into the cells R_1, R_2, …., R_N, and the decoder is completely specified by the codebook, C = (y_1, y_2, …, y_N). The optimality conditions presented next will explicitly determine the optimal partition for a given codebook and the optimal codebook for a given partition.

For a given partition $\{R_i; i = 1, ---, N\}$ the optimal code vectors satisfy centroid condition

$$y_i = \text{cent}(R_i) \tag{4.14}$$

First Proof (*General Case*): The average distortion is given by

$$D = \sum_{i=1}^{N} \int_{Ri} d(x, y_i) f_x(x) dx = \sum_{i=1}^{N} p_i \int d(x, y_i) fx_i(x) dx \tag{4.15}$$

where we use the notation $f_{xi}(x)$ to mean the conditional pdf for x given $x \in R_i$ and where $p_i = P[x \in Ri]$ Since the partition is fixed, each term can be separately minimized by finding y_i that will minimize the expected distortion, or

$$E[d(x, y_i) x \in R_i] = \int d(x, y_i) fx_i(x) dx \tag{4.16}$$

By definition, the centroid y_i = cent(R_i) minimizes this conditional distortion. Centroid of a partition considers summation average of all the samples in the partition. It has the property that it is at the center of the partition. So it will offer minimum difference in the calculation of distortion.

The following two proofs are restricted to the case where the squared error distortion measure is used as the performance measure. They are based on the view that the decoder receives partial information about the input vector and must perform a statistical estimate of the input given the transmitted index. The second proof finds the optimal decoder by finding the optimal (not necessarily linear) estimator of X and the third proof is based on the recognition that the decoder always performs; a linear operation and the optimal estimate is linear. In the second and third proof decoder receives partial information regarding the partition to which input vector belongs in terms of index consisting selector function which is explained below. The decoder is required to make the statistical estimate of the input for the given selector vector. This can be understood if we refer to the second and third proof given below. This is a lossy quantization.

As background for these proofs, we introduce the binary vector S of selector functions in the following treatment for modeling vector quantizers. Since the partition is fixed, the selector functions, $S_i(x) = 1_{Ri(x)}$, are well-defined and the encoder output can be expressed in terms of the binary vector

$$S = (S_1(X), ---, S_N(X)) \qquad (4.17)$$

Where for each i $\in$ *I,* $S_i = S_i(X)$ has value unity when $X \in R_i$, and value zero otherwise. Then the N-dimensional vector $S = (S_1, S_2, --, S_N)$ fully describes the output of the encoder. This vector can only take on values in which all components but one is zero and the remaining component has value unity. With this background, the remaining proofs can be presented.

Second Proof (*Squared Error Distortion):* The decoder can be regarded as a vector-valued function of S, with output Y = F(S) where Y is a k dimensional reproduction vector. To minimize

$$D = E[\|X - F(s)\|^2] \qquad (4.18)$$

$$F(S) = E[X/S] \qquad (4.19)$$

or, in terms of sample values

$$F(z) = E[X/S = z] \qquad (4.20)$$

Now, recognizing that only one component of the binary vector z can be nonzero, there are only N distinct values that can be taken on by z. In particular, suppose $z = u_i$ where u_i has a 1 in the i^{th} coordinate and 0 elsewhere. Then

$$F(u_i) = y_i = E[x|s_i = 1] \qquad (4.21)$$

which again proves the necessity of the centroid condition for optimal quantization.

Third Proof (*Squared Error Distortion*): From the primary decomposition of a vector quantizer, we can express the reproduction vector, **y**, as

$$y = Q(x) = \sum_{j=1}^{N} y_j S_j \tag{4.22}$$

It may be seen that this is a linear combination of the observable random variables Sj. Since we wish to minimize the average distortion $E[\|X - Y\|^2]$, we seek the best estimate of X using the estimator Y, which is constrained to be linear combination of well-defined random variables, S_j .By the orthogonality principle, the condition for optimality is that the estimation error X – Y must be orthogonal to each of the observable variables, S_i. Thus we have

$$E(XSi) = \sum_{j=1}^{N} yjE(XSjSi) \tag{4.23}$$

Note that E(SjSi) is zero for j ≠ i and is equal to E(Si^2) =Pi when j = i. Also the expectation E(XSi) can be expressed in the form

$$E(XSi) = E[X|S_i = 1]Pi \tag{4.24}$$

From these observations equation (4.23) simplifies and yields

$$y_i = E(XSi)/E(Si^2) = E[X| S_i = 1] = cent(R_i) \tag{4.25}$$

This result assumes that the partition is degenerate in the sense that each region has nonzero probability of containing the input vector, i.e., Pi ≠ 0. In the degenerate case where Pi = 0, we have what is called empty cell problem. For such a partition region, the centroid is undefined and clearly it makes no sense to have a code vector dedicated to represent this cell. A variety of heuristic solutions have been proposed for handling the empty cell problem. In some methods a cell is declared “empty” if it has 3 or fewer training vectors. In one method, the cell with the highest number of training vectors is assigned a second code vector by splitting its centroid into two vectors and the empty cell is deleted. In another method the centroid of the cell with the highest partial distortion is split. If c cells are empty in a given iteration, then the c cells with the highest partial distortions can have their centroids split .Empty cell can be merged with nearby cell having highest partial distortions without affecting partial distortion of the nearby cell.

Zero Probability Boundary Condition

A necessary condition for a codebook to be optimal for a given source distribution is

$$P\ (\ U^{N}_{j=1}\ Bj\ ) = 0 \tag{4.26}$$

That is the boundary points occur with zero probability. An alternative way to give this condition is to require that the collection of points equidistant from at least two code words has probability 0, that is,

$$P(\ \mathbf{x} : d(\mathbf{x}, \mathbf{y}_i\ ) = d(\mathbf{x}, \mathbf{y}_j\ ) \text{ for some } i \neq j\ ) = 0 \tag{4.27}$$

Sufficiency of the Optimality Conditions

Supposes one has a vector quantizer that satisfies the centroid condition, the nearest neighbor condition, and the zero-probability boundary condition. The quantizer is *locally optimal* if every small perturbation of the code vectors does not lead to a decrease in D. It is *globally optimal* if there exist no other codebook that gives a lower value of D. If we have a codebook that satisfies both necessary conditions of optimality, it is widely believed that it is indeed locally optimal. No general theoretical derivation of this result has ever been obtained. For the particular case of a discrete input distribution such as a sample distribution produced by training sequence, a vector quantizer satisfying necessary conditions is indeed locally optimal. In the discrete input case, a slight perturbation of a code vector will not alter the partitioning of the set of input vectors as long as none of the input values lies on a partition boundary. Once the partition stays fixed, the perturbation causes the centroid condition to be violated and the average distortion can only increase. At least under these conditions, a vector quantizer that satisfies the necessary conditions will be locally optimal. Locally optimal quantizers can, in fact, be very suboptimal. Code book is prepared based on initial codebook. Optimization of the initial codebook is performed through iterations. Local optimization will improve the codebook which will have the best performance in the given partitions of the initial codebook. So these quantizers are locally optimum. Locally optimum quantizers do not try to optimize the partition other than the partition with the respective code word. So they are not globally optimal or suboptimal. For non uniform sources codebook for the quantization is prepared by Lloyd iteration for codebook improvement until iteration converges. Overload distortion for unbounded input is strongly influenced by the shape of the input pdf. To deal with unbounded input (overload region) one can consider additional support regions (partitions) to the existing partitions. Another way to deal with unbounded input is to limit the input prior to quantization, which is a standard practice.

4.4 Design of Full Search Vector Quantizer

The necessary conditions for optimality provide the basis for iteratively improving a given vector quantizer. If the iteration continues to convergence, a good (may be close to optimal) quantizer can be found. The iteration begins with a vector quantizer consisting of its codebook and the corresponding optimal (NN) partition and then find the new codebook which is optimal for that partition. This new codebook and its NN partition are then a new vector quantizer with average distortion no greater than the original quantizer. Although each of these steps of optimizing a partition for a codebook and a codebook for a partition is simple and straightforward, the simultaneous satisfaction of both conditions is not easy to achieve. There are no closed-form solutions to the problem of optimal quantization. The repeated application of the improvement step, however, yields an iterative algorithm which at least reduces (or leaves unchanged) the average distortion at each step. The terminal condition for the iteration is that if fractional drop in distortion $(D_m – D_{m+1}) / D_m$, is below a suitable threshold.

We begin with the problem of obtaining the initial codebook for improvement since this, too, is a problem of vector quantizer design. In fact, if the initial codebook is good enough, it may not be worth the effort to run further improvement algorithms. There are a variety of techniques for generating a codebook that have been developed in cluster analysis (for pattern recognition) and in vector quantization (for signal compression).We survey several of the most useful.

4.4.1 Random coding

Perhaps the simplest conceptual approach towards filling a codebook of N code words is to randomly select the code words according to the source distribution, which can be viewed as a Monte Carlo design. Selection criteria for training vector are given below. The obvious variation when designing based on a training sequence is to simply select the first N training vectors as code words. If the data is highly correlated, it will likely produce a better codebook. Say if one takes every kth training vector , where k is the vector dimension. This technique has been often used in the pattern recognition literature and was used in the original development of the k-means technique (1). One can be somewhat more sophisticated and randomly generate a codebook using not the input distribution, but the distribution which solves the optimization problem defining Shannon's distortion-rate function. In fact, the Shannon source coding theorems imply that such a random selection will on average yield a good code (68) (3) (4). At the most kN training vectors are required.

4.4.2 Pruning

Pruning refers to the idea of starting with the training set and selectively eliminating (pruning) training vectors as candidate code vectors until a final set of training vector remains as the codebook. In one such method, a sequence of training vectors is used to populate a codebook recursively as follows: put the first training vector in the codebook. Then compute the distortion between the next training vector and the first code word. If it is less than some threshold, continue. If it is greater than the threshold, add the new vector to the codebook as a codeword. With each new training vector, find the nearest neighbor in the codebook. If the resulting distortion is not within some threshold, add the training vector to the codebook. Continue in this fashion until the codebook has enough words. For a given finite set of training vectors, it is possible that the resulting codebook will have fewer than the desired number of code vectors. If this happens, the threshold value must be reduced and the process repeated. A typical choice of threshold value for the MSE distortion measure is proportional to 2^{2r} where r is the rate of the code. This technique is well known in the statistical clustering literature (69). Pruning eliminates some training vectors as candidate code vectors to have distortion less than the threshold, so it makes the quantization better. Pruning in a tree will generate variable length code which can be handled by the algorithms like BFOS. An algorithm, which finds the lower boundary of the convex hull of the rate/distortion pairs of the pruned sub trees of a given tree and which specifies the sub trees that lie on this convex hull is developed by Breiman, Friedman, Olshen and Stone (BFOS algorithm). Brieman et. al. developed an algorithm for optimally pruning a tree trading off the number of leaves with probability of error for a classification tree and mean squared error for a regression tree. General complexity and cost criteria are considered in the work done by Chou et.al. , which is a generalization on BFOS algorithm.

4.4.3 Pair wise Nearest Neighbor Design

A more complicated , but better , means of finding a codebook from a training sequence is the pair wise nearest neighbor (PNN) clustering algorithm proposed by Equitz (6)(7).Similar algorithms have also been used in the clustering literature(70)(9). This is also a form of pruning as it begins with the entire training sequence of L vectors, and ends with a collection of N vectors.

First compute the distortion between all pairs of vectors. The two training vectors having the smallest distortion are combined into a single cluster and represented by their centroid. We now have $L - 1$ clusters, one containing two vectors and the rest containing a single vector. Henceforth at each step clusters may have more than one vector. Suppose now that we have K clusters with $N < K \leq L - 1$ and we wish to merge two of the clusters to get a good set of $K - 1$ clusters. This single step merging can be done in an optimal fashion as follows: For every pair of clusters drawn from the full collection of K clusters, compute the increase in average distortion resulting if the two clusters and their centroids are replaced by the merged two clusters and the corresponding centroid. When the best pair of clusters for merging is found, they are merged to form a codebook with $K - 1$ vectors. Continue in this way until only N vectors remain.

Note that each merge is optimal, but the overall procedure need not be optimal, that is, need not produce the optimal code book of the given size. As a size of training set increases, initial codebook size using PNN also grows, which will consume more time. The solution to this issue can be as follows. The training vectors can be divided into groups. Rest of the groups of L training vectors can be represented by the centroid of the group. PNN algorithm can be applied to first group of training vectors and centroids of the rest of the groups.

4.4.4 Product Codes

In some cases a product code book may provide a good initial guess. For example, if one wishes to design a codebook for a k-dimensional VQ with codebook size 2^{kR} for some integral resolution R, then one can use the product of k scalar quantizers with 2^R words each. Thus if q(x) is a scalar quantizer, then $Q(x_0 , \ldots , x_{k-1}) = (q(x_0 , \ldots , q(x_{k-1}))$, the Cartesian product of the scalar quantizers , is a vector quantizer. This technique will not work if R is not an integer.

4.4.5 Splitting

Linde et al. introduced a technique that resembles the product code initialization in that it grows large codebooks from small ones, but differs in that it does not require an integral number of bits per symbol (10). The method is called splitting algorithm and it produces increasingly larger codebooks of fixed dimension. The globally optimal resolution 0 codebook of a training sequence is the centroid of the entire sequence. The one code word , say $\mathbf{y_0}$, in this codebook can be "split" into two code words , $\mathbf{y_0}$ and $\mathbf{y_0} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a vector of small Euclidean norm. One choice of $\boldsymbol{\epsilon}$ is to make it proportional to the vector whose i^th^ component is the standard deviation of the i^th^ component of the set of training vectors. Another choice is to make it proportional to the eigenvector corresponding to the largest eigen value of the covariance matrix of the training set. This new code book has two words and can be no worse than the previous codebook since it contains previous codebook. The iterative improvement algorithm can be run on this codebook to produce a good resolution 1 code.

When complete, all of the code words in the new codebook can be split, forming an initial guess for a resolution 2 code books. One continues in this manner, using a good resolution r codebook to form an initial resolution r + 1 codebook by splitting. This algorithm provides a complete design technique from scratch on a training sequence, very similar to successive approximation quantizers.

The Generalized Lloyd Algorithm

The generalized Lloyd algorithm for VQ design is sometimes known as the k-means algorithm after MacQueen (1), who studied it as a statistical clustering procedure. The algorithm is based on iterative use of the codebook modification operation, which generalizes the Lloyd iteration for scalar quantization. Following is the generalization for the vector case when the joint pdf of the input vector is assumed to be known and continuously distributed.

The Lloyd Iteration for Codebook Improvement

Known Statistics:

1. Given a codebook , $C_m = \{ \mathbf{y_i} \ ; i=1,.., N\}$, find the optimal partition into quantization cells , that is , use the nearest Neighbor Condition to form the nearest neighbor cells :

$$R_i = \{\mathbf{x}: \mathbf{d(x,y_i)} < \mathbf{d(x,y_j)} \quad ; \text{all } j \neq i \}.$$

If **x** yields a tie for distortion, e.g., if $\mathbf{d(x,y_i)} = \mathbf{d(x,y_j)}$ for one or more $j \neq i$, then assign **x** to the set R_j for which j is smallest.

2. Using the centroid Condition, find $C_{m+1} = \{ \text{cent}(R_i) : i = 1,\ldots,N\}$, the optimal reproduction alphabet (codebook) for the cells just found.

Table 4.3 The Lloyd Iteration for Codebook Improvement Known Statistics

The Lloyd Iteration for Empirical Data
1.Given a codebook , $C_m = \{ \mathbf{y_i} ; i=1,.., N\}$, partition the training set into cluster sets R_i using the nearest Neighbor Condition $R_i = \{\mathbf{x} \in T: \mathbf{d(x,y_i)} < \mathbf{d(x,y_j)} ; \text{all } j \neq i \}$ (and a suitable tie-breaking rule). **2.** Using the centroid Condition, compute the centroids for the cluster sets just found to obtain the new codebook , $C_{m+1} = \{ \text{cent}(R_i) : i = 1,\dots,N\}$. If an empty cell was generated in step (a) , an alternate code vector assignment is made (in place of the centroid computation) for that cell.

Table 4.4 The Lloyd Iteration for Empirical Data

The Lloyd iteration can be applied to the discrete input distribution defined from the training set *T* to obtain a locally optimal quantizer for this distribution. Lloyd iteration assigns the input to a partition using nearest neighbor condition. So it does not worry about the irregular / odd shaped regions in multidimensional space. In general after the application of Lloyd iteration, code books and hence partitions represent irregular / odd shaped regions. This does not make the algorithm impractical as we have to deal with the code vectors and not the partitions.

The Generalized Lloyd Algorithm
1. Begin with an initial codebook C_1 . Set m =1 . **2**. Given a codebook , C_m = { $\mathbf{y_i}$; i=1,.., N} , perform the Lloyd Iteration to generate the improved codebook C_{m+1} . **3**. Compute the average distortion for C_{m+1} . If it has changed by a small enough amount since the last iteration , stop. Otherwise set m + 1 → m and go to Step 2.

Table 4.5 The Generalized Lloyd Algorithm

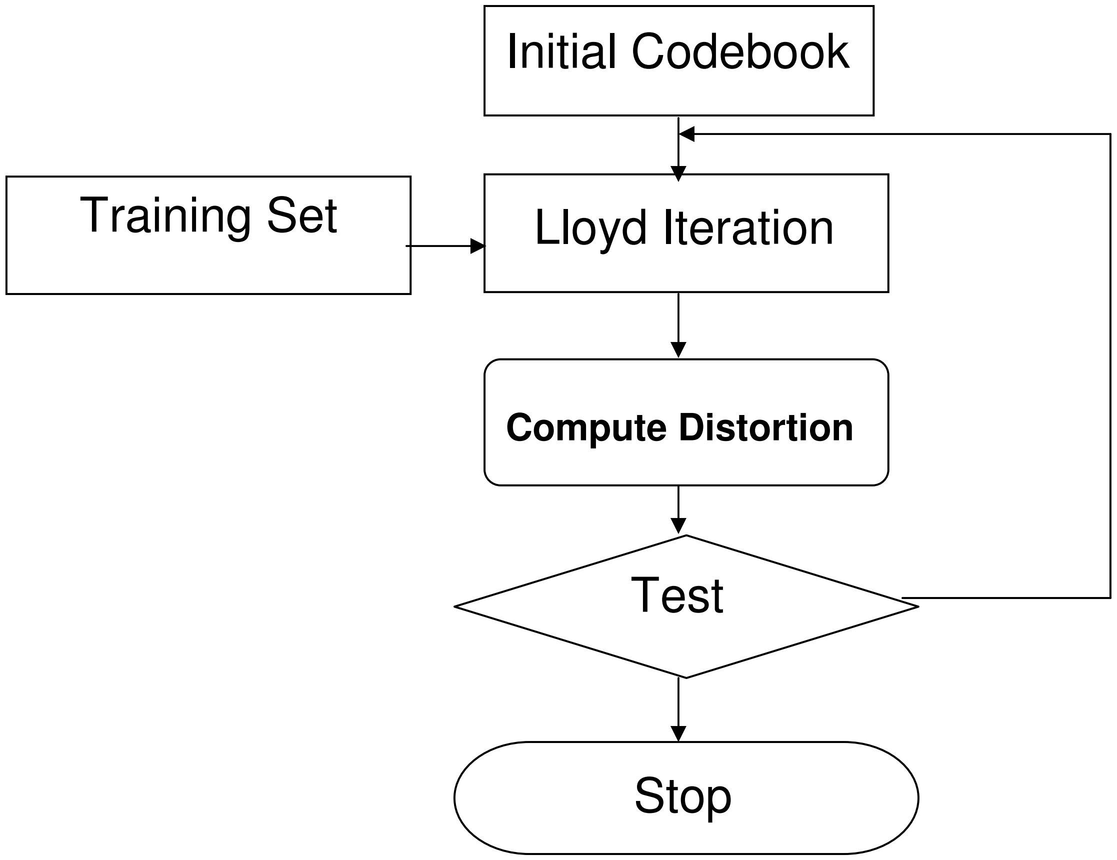


Figure 4.2 Lloyd Algorithm flow chart

The flow chart for the generalized Lloyd algorithm is shown in Figure 4.2. If an empty is generated in GLA, an alternative code vector assignment is made (in place of the centroid computation for that cell.

4.4.6 Stochastic Relaxation

By introducing randomness into each iteration of the GL algorithm it becomes possible to evade local minima , reduce or eliminate the dependence of the solution on the initial codebook , and locate a solution that may actually be a global minimum of the average distortion as a function of the codebook. A family of optimization technique called stochastic relaxation (SR) is characterized by the common feature that each iteration of a search for the minimum of a cost function (e.g., the average distortion) consists of perturbing the state, the set of independent variables of the cost function (e.g., the codebook) in a random fashion. The magnitude of the perturbation generally decreases with time, so that convergence is achieved. The earliest use of randomness for VQ design was proposed in (10) where noise is added to the training vectors prior to a Lloyd iteration and the variance of the noise is gradually decreased to zero. This is indeed an SR algorithm although it was not so labeled.

4.4.7 Stimulated Annealing

Simulated annealing is a stochastic relaxation technique in which a randomly generated perturbation to the state (the codebook) at each iteration is accepted or rejected probabilistically where the probability depends on the change in value of the cost function resulting from such perturbation.

4.4.8 Fuzzy Clustering

In designing a VQ from empirical data, selector functions are encountered. These functions may be regarded as membership functions. A cluster is said to be fuzzy set if we may assign to each element of the training set a degree of membership or partial membership value between zero and one which indicates to what extent the particular vector is to be regarded as belonging to that set. Based on fuzzy distortion and the degree of fuzziness VQ codebook is obtained.

4.4.9 Improvement of Codebook with Gaussian pdf over Codebook with Uniform pdf

$$A_G = \frac{1}{\sqrt{2\Pi}} \int_{-1}^{1} x e^{-x^2/2} dx \qquad (4.28)$$

(For zero mean)

As Gaussian pdf is also even symmetric function

$$A_G = \frac{2}{\sqrt{2\Pi}} \int_{0}^{1} x e^{-x^2/2} dx \qquad (4.29)$$

Assuming $x^2/2 = y$, we have

$xdx = dy$

for limit x=1, y=1/2 and for x=0, y=0.

$$A_G = \frac{2}{\sqrt{2\Pi}} \int_{0}^{1/2} e^{-y} dy \qquad (4.30)$$

$$A_G = \frac{2}{\sqrt{2\Pi}} \left[-e^{-y}\right]_0^{1/2} \qquad (4.31)$$

$= \frac{2}{\sqrt{2\Pi}}\left[1 - \frac{1}{1.6486}\right]$

$= \frac{2}{\sqrt{2\Pi}}(0.39342)$

= 0.3139

Area difference = A_U-A_G = 1-0.3147 = 0.6861 is equivalent to 20 log 0.6861 = 3.27 dB. This is the improvement if we apply codebook with Gaussian pdf instead of uniform pdf

4.5 Full search VQ Results

In the following section Lloyd iteration for empirical data (training set) has been applied to generate codebook. Codebook for uniform and Gaussian pdf is generated .The FSVQ is applied to speech and image. The number of code words is 32.

Table 4.6 Wave file on which FSVQ applied

Wave file Properties	**om.wav**
Bitrate	64 kbps
Audio Sample Size	8 - bit
Channel	1 (mono)
Audio Sample rate	8 kHz
Audio format	PCM

The Signal to Quantization Noise Ratio (SQNR) is defined by normalizing the signal power by the quantization error power and taking a scaled logarithm:

$$SQNR = 10\log_{10}\left(\sigma^2 / MSE\right) \qquad (4.32)$$

, measured in dB .It is usually assumed that the input process has zero mean and hence that the signal power is same as the variance σ^2 of the signal. MSE is Mean Squared Error where error is $\varepsilon = Q(X) - X$

Looking to the results in Table 4.7 SQNR with Gaussian pdf is better compared to SQNR with Uniform pdf for any VQ dimension. As VQ dimension increases SQNR decreases because more samples are grouped, this will increase quantization error.

Table 4.7 SQNR (dB) for VQ dimensions for Om.wav

VQ dimension	**SQNR (dB) Uniform pdf**	**SQNR (dB) Gaussian pdf**
1	15.544	19.947
2	14.355	16.452
3	7.6393	11.21
4	8.2723	10.78
5	6.9166	9.9534
6	7.4164	9.8926
7	6.8965	9.3647
8	6.0186	8.7807
9	6.86	8.6589
10	6.6571	8.2975

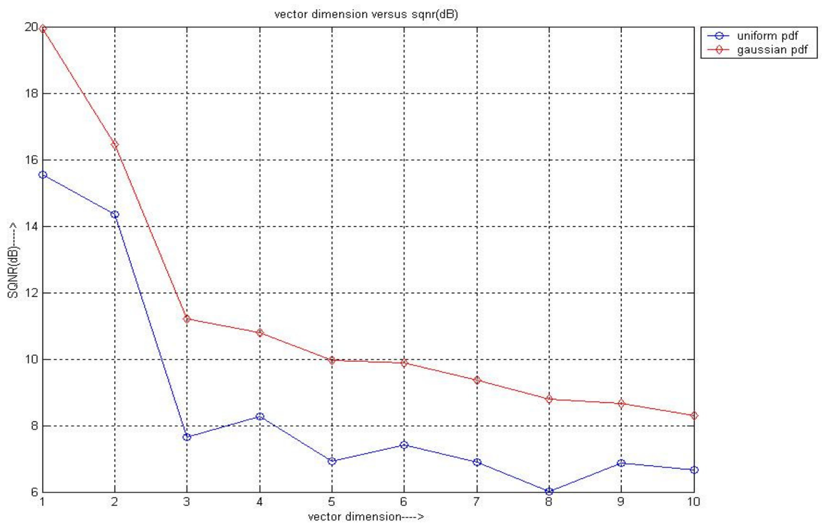


Figure 4.3 Vector dimension versus SQNR(dB) for om.wav

Full search VQ applied on image

Codebook design is very complex for VQ applied on image. Code words are blocks of pixels, usually power of 2. For 4x4 blocks at 1 bpp 2^{16} code words are there.We take 16 images of size 256 x 256, i.e. 2^{16} training vectors (4x 4 each).Codebook size becomes 2^{16} x 4 x 4 x 8 bits = 8.3 Mbits . If we consider 4x4 blocks at 0.5 bpp , we have 2^{8} ,i.e.256 code words. We take one 256x256 image which consists of : 4096 training vectors. Now codebook size becomes 256 x 4 x 4 x 8 bits = 32.8 Kbits.

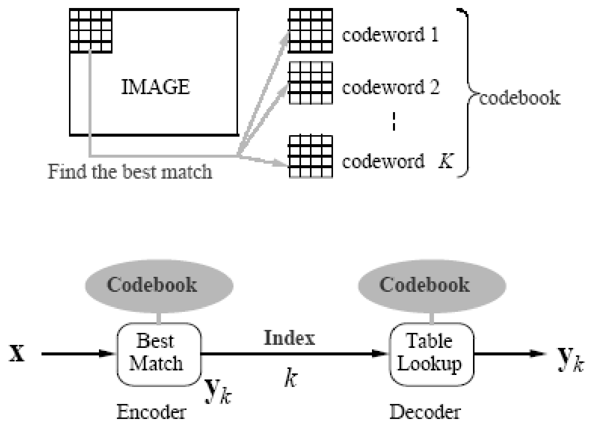


Figure 4.4 Vector Quantization of an image

In the following application codebook is designed from training image. 4x4 blocks are used at 0.5 bpp: (2 to the power 8) i.e. 256 code words .Training image is 256x256 image, i.e. 4096 training vectors. Codebook size becomes 256 x 4 x 4 x 8 bits = 32.8 Kbits .cameraman.tif file is test file to be VQ encoded , SQNR comes out to be -23.844 dB.

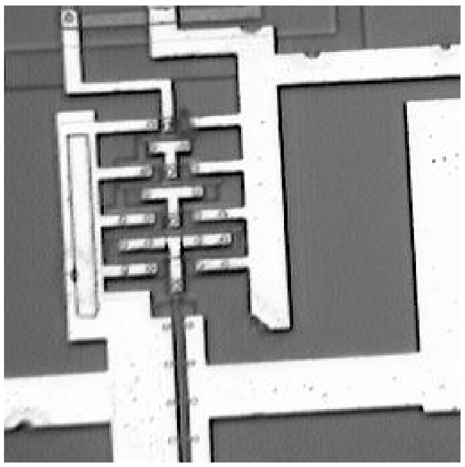
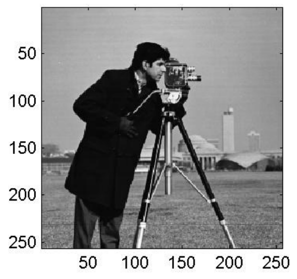


Figure 4.5 Training image for code book generation



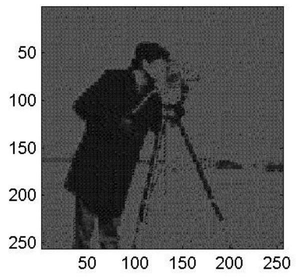


Figure 4.6 Test Image before and after FSVQ applied

If we vary the number of code words the results are as follows. As number of code words varies, bits per pixel vary. Codebook size is number of code words x 4 x 4 x 8 bits = Kbits

	No.of codewords (4 x 4 block) =16	No.of codewords (4 x 4 block) =32	No.of codewords (4 x 4 block)= 256
SQNR(dB) (Uniform pdf)	-21.635	-19.217	-20.241
SQNR(dB) (Gaussian pdf)	-25.702	-25.024	-23.372

Table 4.8 SQNR (dB) for different number of code words for test image (4 x 4 block)

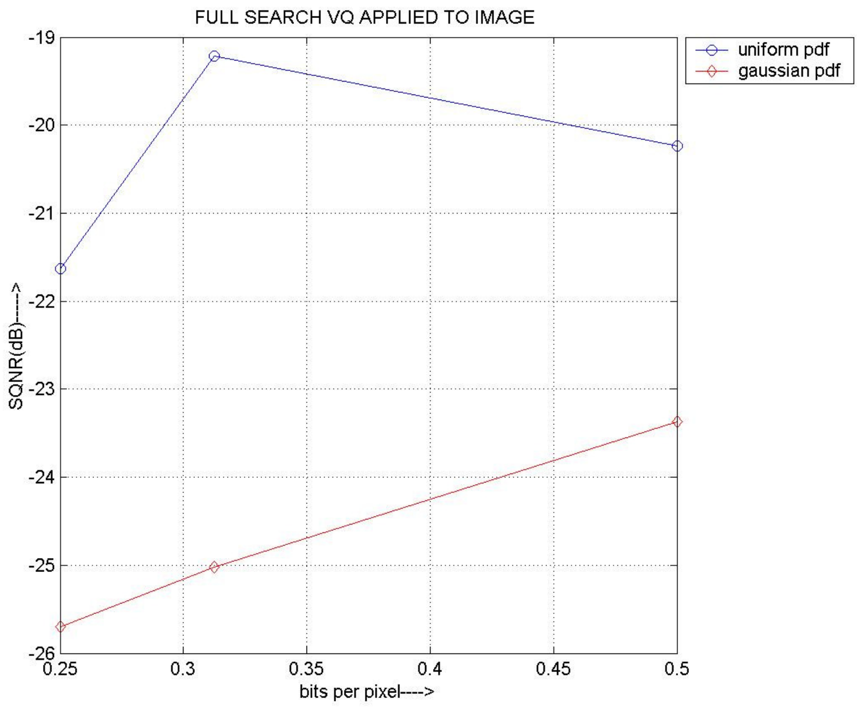


Figure 4.7 SQNR(dB) Versus Bits per pixel for Test Image (4 x 4 block)

	No.of codewords (8 x 8 block) =16	No.of codewords (8 x 8 block) =32	No.of codewords (8 x 8 block)= 256
SQNR(dB) (Uniform pdf)	-27.435	-26.74	-24.229
SQNR(dB) (Gaussian pdf)	-29.267	-29.132	-28.642

Table 4.9 SQNR (dB) for different number of code words for test image (8 x 8 block)

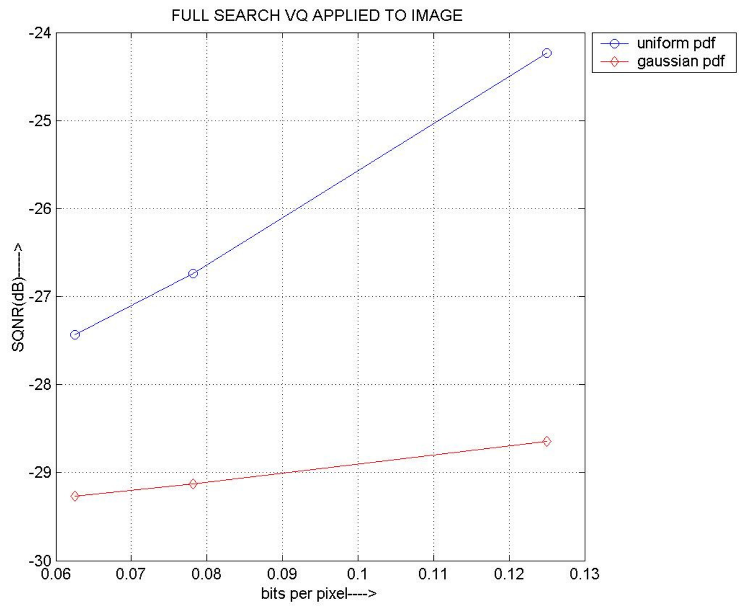


Figure 4.8 SQNR(dB) Versus Bits per pixel for Test Image (8 x 8 block)

	No.of codewords (16 x 16 block) =16	No.of codewords (16 x 16 block) =32	No.of codewords (16 x 16 block)= 256
SQNR(dB) (Uniform pdf)	-31.852	-31.687	-30.808
SQNR(dB) (Gaussian pdf)	-32.671	-32.628	-32.409

Table 4.10 SQNR (dB) for different number of code words for test image (16 x 16 block)

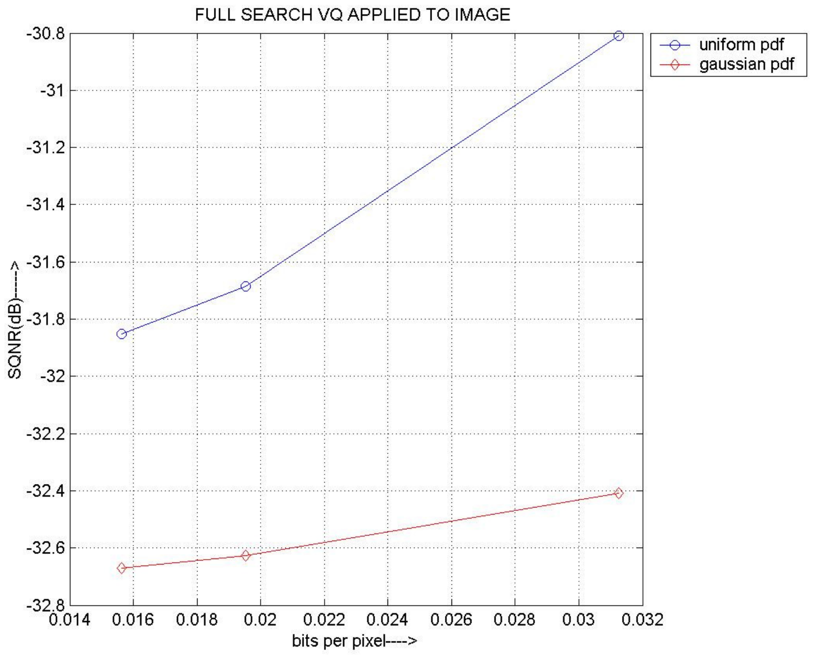


Figure 4.9 SQNR(dB) Versus Bits per pixel for Test Image (16 x 16 block)

4.6 Summary

This chapter described the definition and concept of FSVQ. It also includes primary structure of vector quantizer and Nearest Neighbors quantizer. As a performance measure criteria distortion calculation is explained. Optimal conditions for VQ are studied. FSVQ design is applied on speech. SQNR for various VQ dimension are obtained. Similarly FSVQ design is applied to image. SQNR for 4x4, 8x8 and 16x16 blocks are obtained, which are depicted in SQNR versus bpp graph. For speech signal, applying FSVQ yields better SQNR with Gaussian pdf compared to uniform pdf .Applying FSVQ on an image, as we increase bits per pixel , SQNR increases for Gaussian pdf as well as uniform pdf for 4x4 block , 8x8 block and 16x16 block. In case of images uniform distribution is giving better results than Gaussian because pixel distribution is more close to uniform rather than Gaussian. Training image is selected in such away that it has almost all types of gray level variation to offer variety of codewords.There is no selection criteria for test image. If training and test image is same then we get best performance with zero quantization error.

Chapter 5

Tree Structured Vector Quantizer

5.1 Tree Structure

One of the most effective and widely-used techniques for reducing the search complexity in VQ is to use a tree-structured codebook search. In tree-structured VQ (TSVQ), the search is performed in stages. In each stage a substantial subset of candidate code vectors is eliminated by a relatively small number of operations. In an m-ary tree search with balanced tree, the input code vector is compared with m predesigned test vectors at each stage or node of the tree. The nearest (minimum distortion) test vector determines which of m paths through the tree to select in order to reach the next stage of testing. At each stage the number of candidate code vectors is reduced to 1/m of the previous set of candidates. Such a tree-structured search is a special case of classification tree (22). In many applications m=2 and we have a binary tree.

If the codebook size is $N=m^d$, then d m-ary search stages are needed to locate the chosen code vector. An m-ary tree with d stages is said to have breadth m and depth d. The basic structure of the tree search is shown in figure 5.1, where each node in the tree (except for the terminal nodes) is associated with asset of m test vectors, forming a node codebook. The figure shows the tree to the depth of three. The encoder first searches the root codebook C* and finds index i, then i becomes the first symbol in the m-ary channel vector and encoder advances along the i^{th} branch emanating from the root node to node (i), where each node is represented by the m-ary path map describing how the encoder went from the root node to the current node (i). Given that the search has reached the current node (i), the encoder searches the corresponding node codebook C_i to find the minimum distortion test vector, which determines the next branch, and so on. Distributions of output points on the tree will decide the path through the tree for encoding. If distribution of output points on the tree is not symmetrical, then the tree becomes unbalanced.

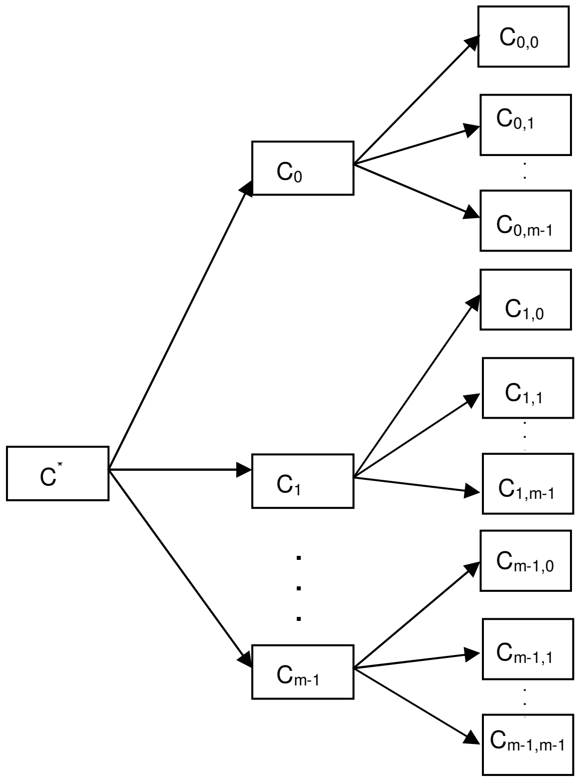


Figure :- 5.1 Basic Structure of Tree-Structured VQ Encoding

Since a tree-structured codebook only affects the search strategy, the decoder does not need the test vectors and is in fact identical to that of the conventional VQ.

TSVQ Encoder
0. Given : Depth d, breadth m, rate $\log_2 m^d$ bits per input vector **x.**
1. Root node: Find the code word $\mathbf{y} \in \mathbf{C}^*$ minimizing d(**x,y**). Let $u_0 \in \{ 0,1,\ldots,m-1\}$ be the index of this minimum distortion word. Set the one-dimensional channel m-ary code word to $u^1 = u_0$ and advance to node (u^1). Set the current tree depth k=1.
2. Given the k-dimensional channel codeword $u^k = (u_0 , u_1,\ldots,u_{k-1})$, and the current node (u^k) , find the code word $\mathbf{y} \in \mathbf{C}_\mathbf{u}^\mathbf{k}$ minimizing the distortion d(**x,y**) . Let u^k denote the index of the minimum distortion code word. Set the (k+1)-dimensional channel m-ary codeword u^{k+1} equal to the concatenation of u^k and u_k : $u^{k+1} = (u^k , u_k) = (u_0 , u_1,\ldots,u_k)$
3. If k +1 =d , halt with the final channel codeword u^d (corresponding to a reproduction vector $\mathbf{C}_\mathbf{u}^{\mathbf{d-1}}$). Otherwise set k +1 $\rightarrow$ k and go to step 2.

Table 5.1 TSVQ Encoder

The number of search operations at each stage is proportional to m since each test is an exhaustive nearest neighbor search through a set of m test vectors. Thus , the total search complexity is proportional to md rather than m^d where the proportionality constant depends on the vector dimension k. On the other hand, the storage requirement of TSVQ is increased compared to unstructured VQ. In addition to storing m^d code vectors (the leaves of the tree) , the test vectors for each node of the tree must also be stored. There is one node at the first stage , m nodes at second stage , m^2 nodes at the third stage, etc. Hence, the total number of nonterminal nodes is $1 + m + m^2 + \ldots + m^{d-1} = (m^d - 1) / (m -1)$. Since each non terminal node stores m code or test vectors , the total number of k-dimensional vectors to be stored including code vectors is $m(m^d - 1) / (m -1)$. Alternatively, one must store m^d code vectors plus

$$\sum_{l=1}^{d-1} m^{l} = m(m^{d-1} - 1) / (m - 1) \tag{5.1}$$

additional test vectors.

The simplest tree structure to implement is the binary tree where m=2. In this case the complexity is reduced by a factor of $2d/2^{d}$ compared to ordinary VQ and the total storage is $2(2^{d} - 1)$ vectors , slightly less than double that of ordinary VQ with the same codebook size N = 2d . For a given codebook size the complexity increases proportional to m/logm and the storage decreases in proportion to m/ (m-1) as the tree breadth m increases. For a binary tree, the complexity is lowest and the storage is highest. In TSVQ as we increase m, complexity increases and storage reduces. Storage and distortion requirements can be compensated by increasing the depth of the tree with the increase in m. Alternately the tree structure can also be formed as below.

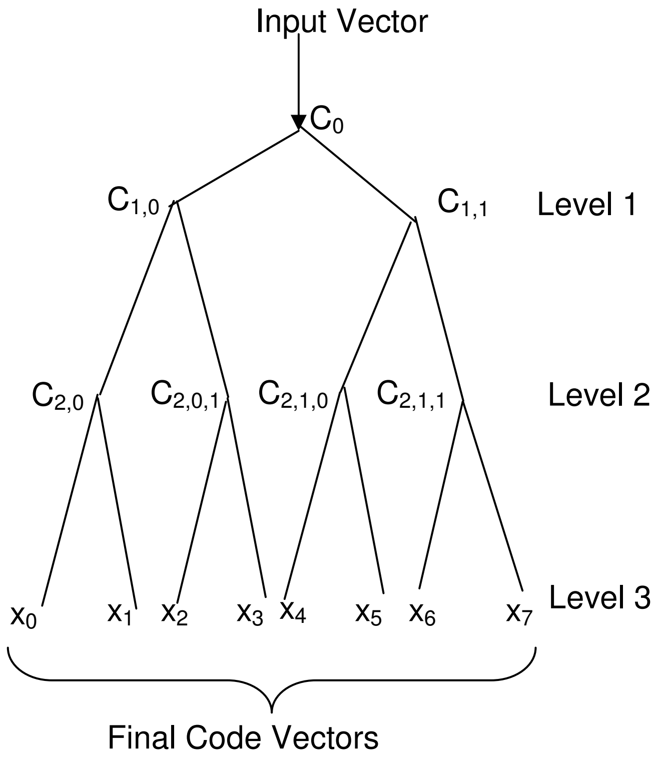


Figure 5.2 Tree Structure VQ with *L*=3 levels.

In the codebook C, at the root of the tree is a single codebook $\boldsymbol{C_0}$ of length m and size $M_0=2^{mR_0}$. This situation is depicted in figure 5.2. Here the input vector is used to calculate the codebook for various VQ dimension and tree depth(L).

At level 1 of the tree, there are M_0 codebooks $\boldsymbol{C_{1.q0}}$ where q_0 =1,2 ……, M_0 -1.Each of these codebook is of length m and size $M_1=2^{mR_1}$. At level 2 of tree, there are M1 denoted by codebooks $\boldsymbol{C_{2.q0,q1}}$ where q_1 =1,2,3,..,M_0 -1.Each of these codebook is of length m and size $M_2=2^{mR_2}$. The union of all codebooks at the final level (tree depth) forms the codebook of the TSVQ.

In this figure, there are three levels with $M_0= M_1 = M_2=2$. At level 0 (or the root) of the tree is a single codebook $\boldsymbol{C_0}$ size of one .The possible index selection is shown as 1 and 2. At level 1 of the tree, there are two codebooks $\boldsymbol{C_{1.0}}$ $\boldsymbol{C_{1.1}}$, each of size two, for a total of four code words. At level 2 of the tree, there are four codebooks $\boldsymbol{C_{2.0,0}}$, $\boldsymbol{C_{2.0,1}}$, $\boldsymbol{C_{2.1,0}}$ $\boldsymbol{C_{2.1,1}}$, each of size two, for a total of eight code words. At this level leaves are added to form the codebook of TSVQ.

An m-ary tree with L stages is said to breadth m and depth L. The following table shows the codebook size of the TSVQ.

Tree Level (L)	**Codebook Size (M)**	**No. of code words**
Root	1	1
1	$M_0=2^{mR_0}$	$2^{m(R_0)}$ =2
2	$M_1=2^{mR_1}$	$2^{m(R_0+R_1)}$ =4
3	$M_2=2^{mR_2}$	$2^{m(R_0+R_1+R_2)}$ =8
n	$M_n=2^{mR_n}$	$2^{m(R_0+R_1+R_2+\dots\dots+R_n)}$

Table 5.2 Sizes of Codebook and Code words for TSVQ.

A TSVQ is most easily visualized as a tree, which is labeled with vectors and is searched by the encoder. A sequential encoding algorithm is suitable. The encoder can easily be implemented as a finite state machine with one-step memory, with each step the encoder does a full search of a small codebook, produces a binary vector, and then selects a new codebook based on decision.

The rate of resulting (R_1,..,R_n) TSVQ is given by R=1/m * log $_2$ M. Since one codebook must be searched at each level, the search complexity of TSVQ is proportional to$2^{m(R_0 + R_1 + R_2 + ...+R_n)}$. The smallest complexity is achieved for the binary tree with M_n=2 Or R_n=1/m. Total search complexity is proportional to n $2^{mR/n}$. The complexity of TSVQ can grow only linearly in rate and dimension (as opposed to exponentially as in the full search case). But the storage space required for the code words is roughly doubles.

5.2 Design of Tree-Structured VQ

A standard method for designing the tree structure is based on application of the generalized Lloyd algorithm (GLA) to successive stages using a training set. The procedure, which was proposed in (10), is a variation on the standard splitting method of designing full search VQ. Similar methods have been used in the field of pattern classification. It is better to design vector quantizer within the framework of the tree structure rather than imposing the tree structure on the quantizer, because complexity is reduced.

The first step is identical to that of a standard Lloyd design with the splitting technique: the optimum resolution 0 bit code is found, that is the code having only a single codeword, the centroid of the entire training set. This codeword is then split into two, and the Lloyd algorithm is run to produce a good resolution 1 bit code. The ordinary Lloyd (full search) design would now proceed by splitting the two code words and running the Lloyd algorithm on the full training set to obtain a new codebook of four words. Instead, here the nearest neighbor (optimum) encoder is replaced by the following encoder. The encoder first looks at the 1 bit codebook and selects nearest neighbor. Given that selection, it then confines its search to the descendants of the selected codeword, i.e., to the two words formed by splitting the selected word. It encodes the training set and then replaces all four code words by the centroid of all training vectors mapped into those nodes.

The replacement encoder has substituted two binary searches for a single quaternary search, but the remainder of the algorithm is the same. This depth 2 binary tree offers no saving in complexity as the two encoders have comparable complexity and the double binary search is not an optimum search (may not find nearest neighbor). It has following useful property: the first channel codeword bit itself specifies a 1 bit reconstruction. The second bit improves upon the first. The ordinary full search VQ has no such property.

TSVQ design algorithm can be viewed as a simple variant of splitting version of the generalized Lloyd algorithm.

TSVQ Design Procedure
1. Use the training set *T* to generate a code book *C** of size m test vectors for the root node (level 1) of the tree. Partition the training set into m new subsets T_0, T_1, .. , T_{m-1} .
2. For each i, design a test codebook C_i of size m using the GLA applied to T_i (level 2).
3. Partition each training set T_i into m training subsets T_{ij} and use these new training sets to design m^2 test codebooks C_{ij} for level 3
4. Continue this process until level d-1 is reached. The collection of all these test vectors at this level constitutes the codebook.

Table 5.3 TSVQ Design Procedure

5.2.1 Tree Generation

The first step is to compute the centroid of the set of training vectors and use it as the root level codeword. To find the children of this root, the centroid and a perturbed centroid are chosen as initial child code words. For this $\epsilon \geq 0$ is a distortion threshold, $\delta > 0$ is a perturbation, is used in the node splitting technique to generate tree. Based on node splitting technique and a decision criterion, codebook is prepared. This process continuous till pre-selected size of tree depth (*L*) is obtained. This defines the size of the codebook. The tree-generated codebook is used by TSVQ for quantization purpose. The final codebook is the collection of the leaf level code words at tree depth (*L*). The typical values for Є =0.005 and δ =0.01. The perturbation criteria are based on the source distribution. If the samples are highly correlated then small perturbation will do. If the samples have less correlation then perturbation can be high. TSVQ design process deviates from the splitting process of Full Search VQ in the following way. Splitting in Full search VQ considers the entire training set on the calculation of the centroid to generate next resolution code words. While in TSVQ after splitting the code words only the nearest neighbor training vectors of the split code words are considered for the generation of the next resolution code words. Pruning refers to the idea of starting with the training set and selectively eliminating (pruning) training vectors as code vectors until a final set of the training vectors remains as the code book. A sequence of training vectors is used to populate a codebook recursively as follows: put the first training vector in the codebook. Then compute the distortion between the next training vector and the first code word. If it is greater than the threshold, add the new training vector to the codebook as code word. With each new training vector, find the nearest neighbor in the codebook. If the resulting distortion is not within some threshold, add the new training vector to the codebook. Continue in this fashion until the codebook has enough code words. So looking to the working of pruning, it does not generate tree structured code words. Splitting generates tree structured code words. So splitting is used rather than pruning in TSVQ.

5.2.2 Tree search

The tree search must generate various tentative paths in the code tree; compare them to the sequence of speech/image samples, and evaluating an final output path. The encoder rate is $\log_2 b$ bits per sample, where b is the no. of quantizer levels, or branching factor of the tree. For this the breadth-first scheme, meaning that it views all branches that it will ever view at one tree level before moving into the next level. At The decoder, a replica of the code tree generator simply recalculates the sequence chosen at the decoder.

A particular search leads to a particular path down the tree till a terminal node is reached. Each terminal corresponds to a particular vector in a codebook. Binary tree search VQs is also used for design of fixed sequence training vector.

Unlike full searching, the result codeword may not be the optimal one since only part of the tree is traversed. However, the result codeword is usually close to the optimal solution, and the computation is more efficient than full searching. If the tree is reasonably balanced (this can be enforced in the algorithm), a single search with codebook size can be achieved in time, which is much faster than exhaustive searching with linear time complexity.

5.3 TSVQ Results

TSVQ is applied to speech waveform. Extensive computer simulations were carried out to evaluate the performance of each design. Simulation code used a locally generated uncoded speech file maja_ma_ne.wav as, training sequence of 8 kHz-sampled speech. Vectors are set as per VQ dimension and tree depth (L) is considered from 1 to 10. To generate tree, node-splitting criteria is applied. *Tree- structured* codebook was used to reduce the encoder and decoder complexity with large size codebooks. Another uncoded speech file **kem_chho.wav** of 8 kHz-sampled speech is used as test file. To calculate distortion measure, mean squared error was used. From this SQNR was find out to measure the performance of VQ. Also, the same process was repeated using Gaussian pdf. The test speech file was replaced and played without and with Gaussian pdf generated code words to observe the effect of VQ.

TSVQ is constrained since every input vector is constrained to search with the code words at the nodes of the tree rather than searching the entire code book. To simulate the TSVQ, a training sequence and a test speech is required. Table 5.4 shows detailed parameters of both the speech signals.

Speech file/Properties	**Maja_ma_ne.wav**	**Kem_chho.wav**
Use	Codebook Generation	Test File
Bit Rate	64kbps	64kbps
Audio Sample Size	8-bit	8-bit
Channels	1(Mono)	1(Mono)
Audio Sample Rate	8 kHz	8 kHz
Audio format	PCM	PCM
Size	11.7 KB (12,060 bytes)	7.87 KB (8,060 bytes)

Table 5.4 Speech Signals used for TSVQ Simulation

Vectors of training sequence are set as per VQ dimension. To prepare the tree-structured codebook it is essential to generate tree structure. Tree depth (*L*) is a crucial factor to define the size of codebook. For SQNR calculation range of *L* is kept from 1 to 10,while for replacement file the value of *L* is required to be decided by the user. The root (centroid(s)) of the tree is calculated from the samples of training sequence.

For VQ=1 i.e. SQ, at the root of the tree is single centroid of length 1 and size 1.To generate tree, *node- splitting criteria* is applied. At each root and every sub-root two branches are formed. Є=0.01 is used as the splitting factor to traverse the tree. At level 1 of the tree, from centroid's value, value of Є is added in the right side and value of Є is subtracted in the left side. This process is continued till last tree depth is reached. At level 1 of the tree, the size of codebook is two. At level 2 of the tree, the size of codebook is two. At level 3 of the tree, the size of codebook is three and so on.

For VQ=2, at the root of the tree are two centroids of length 1 and size 1.Tree will be generated for both the centroids as explained above. At level 1 of the tree, the size of codebook is four, two for each centroid. At level 2 of the tree, the size of codebook is four, two for each centroid. At level 3 of the tree, the size of codebook is eight, four for each centroid and so on.

To generalize the codebook concept, for VQ=N, at the root of the tree are N centroids of length 1 and size 1.Tree will be generated for all N centroids as explained above. At level 1 of the tree, the size of codebook is 2^1 x N, 1(2^1 x N) for each centroid. At level 2 of the tree, the size of codebook is 2^2 x N, 2(2^2 x N) for each centroid. At level *L* of the tree, the size of codebook is 2^L x N, L (2^2 x N) for each centroid and so on.

Tree-structured codebook is generated for VQ up to 10. For SQNR calculation range of tree depth *L* is kept from 1 to 10, while for replacement file the value of tree depth *L* is required to be decided by the user.

The performance of a compressed sound using TSVQ is also measured with Signal to Quantization Noise Ratio (SQNR). The SQNR defined by normalizing the signal power by the quantization error power and taking a scaled logarithm: f

$$SQNR = 10\log_{10}\left(\sigma^2 / MSE\right) \tag{5.2}$$

measured in dB .It is usually assumed that the input process has zero mean and hence that the signal power is same as the variance σ^2 of the signal. Using wav files as stated in Table 5.4, we obtained the result of SQNR vs Tree depth (*L*) for different VQ dimension without Gaussian pdf. The following figure 5.3 depicts resultant plot for VQ dimensions 1 to10 and tree depth 1 to 10. Table 5.5 shows the values of SQNR in dB for speech without Gaussian pdf. Looking to the results in figure 5.3, with VQ dimension =10 SQNR remains almost constant at different tree depth. The reason is as follows. At such a high vector dimension, the vector components themselves are highly correlated so that MSE remains almost constant at different tree depths. Hence SQNR also remains almost constant at different tree depths.

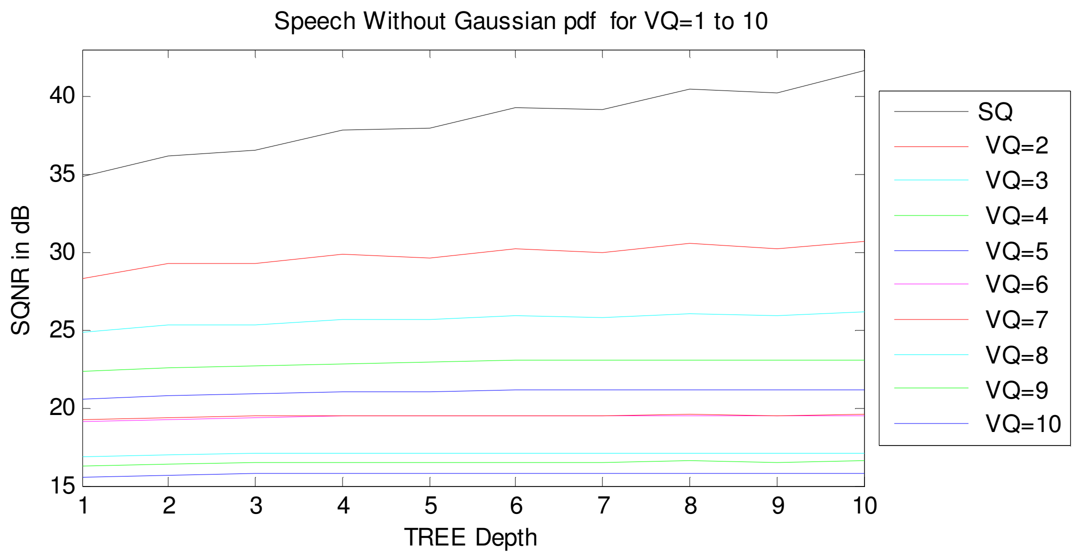


Figure 5.3 TSVQ Encoded Speech With Uniform pdf for VQ=1 to 10

Tree Depth (*L*)	VQ Dimension									
	1(SQ)	2	3	4	5	6	7	8	9	10
1	34.794	28.345	24.782	22.31	20.513	19.071	19.197	16.887	16.232	15.546
2	36.184	29.214	25.312	22.573	20.764	19.276	19.373	17.025	16.403	15.665
3	36.461	29.184	25.34	22.736	20.895	19.369	19.437	17.093	16.475	15.725
4	37.876	29.843	25.708	22.85	20.991	19.448	19.496	17.121	16.523	15.744
5	37.915	29.649	25.626	22.927	21.038	19.47	19.5	17.136	16.53	15.754
6	39.264	30.231	25.928	22.986	21.084	19.505	19.526	17.139	16.547	15.754
7	39.123	29.957	25.792	23.022	21.095	19.501	19.512	17.143	16.539	15.756
8	40.497	30.5	26.061	23.052	21.118	19.52	19.531	17.142	16.549	15.755
9	40.202	30.174	25.892	23.066	21.114	19.509	19.514	17.145	16.54	15.756
10	41.59	30.693	26.138	23.081	21.128	19.524	19.531	17.142	16.55	15.755

Table 5.5 Simulation results of SQNR (dB) for Speech without Gaussian pdf signals.

Using wav files as stated in Table 5.4, we obtained the result, SQNR vs Tree depth (*L*) for different VQ dimension with Gaussian pdf. The following figure 5.4 depicts resultant plot for SQNR vs Tree Depth for VQ dimensions 1 to10 and Tree depth 1 to 10. Table 5.6 shows the values of SQNR in d for speech with Gaussian pdf.

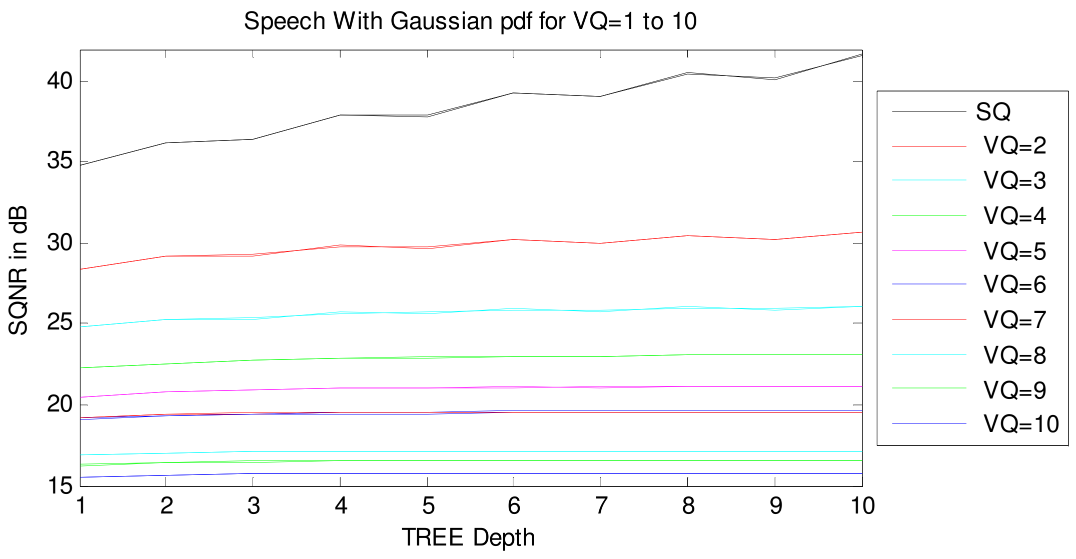


Figure 5.4 TSVQ Encoded Speech With Gaussian pdf for VQ=1 to 10

Tree Depth (L)	*VQ Dimension*									
	1(SQ)	2	3	4	5	6	7	8	9	10
1	34.778	28.415	24.827	22.304	20.493	19.157	19.238	16.882	16.284	15.537
2	36.240	29.180	25.245	22.582	20.760	19.363	19.421	17.033	16.452	15.666
3	36.444	29.249	25.404	22.748	20.881	19.467	19.486	17.099	16.535	15.721
4	37.921	29.816	25.645	22.864	20.996	19.540	19.552	17.137	16.578	15.750
5	37.842	29.724	25.694	22.942	21.029	19.571	19.553	17.146	16.593	15.755
6	39.310	30.206	25.867	23.002	21.091	19.599	19.583	17.155	16.604	15.763
7	39.048	30.036	25.864	23.039	21.087	19.603	19.566	17.153	16.603	15.759
8	40.550	30.475	26.001	23.070	21.125	19.615	19.588	17.158	16.607	15.764
9	40.117	30.257	25.967	23.085	21.107	19.611	19.568	17.154	16.605	15.759
10	41.673	30.670	26.079	23.099	21.136	19.619	19.588	17.158	16.607	15.764

Table 5.6 Simulation results of SQNR in dB for TSVQ Encoded Speech with Gaussian pdf signals.

To check the improvement in performance, SQNR using Gaussian Pdf, Figure 5.5 shows resultant plot for SQNR vs Tree Depth for VQ dimension of 6 and tree depth of 10.

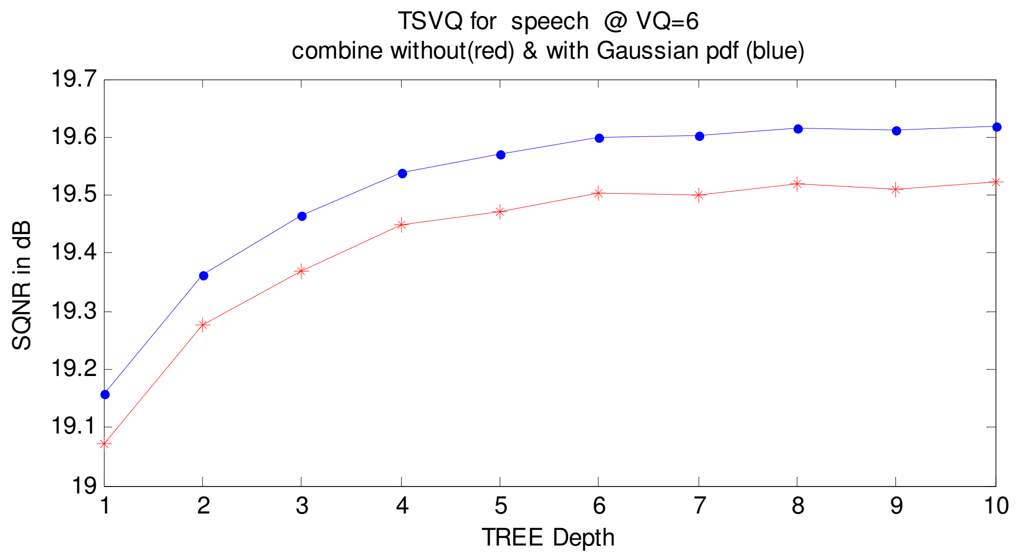


Figure 5.5. TSVQ Encoded Speech Performance with Tree Depths for Uniform and Gaussian pdf and VQ = 6

Now we describe a variety of design and simulations for TSVQ applied to image. Extensive computer simulations were carried out to evaluate the performance of each design. Simulation code used a black and white image 'cameraman.tif' as the training sequence of size 256 X 256. Vectors are set as per VQ dimension and tree depth (*L*) is considered from 1 to 10.

To generate tree, node-splitting criteria is applied. *Tree- structured* codebook is used to reduce the encoder and decoder complexity with large size codebooks. Another black and white image 'moon.tif' is used as a test file. To calculate distortion measure, mean squared error was used. From this PSNR was find out to measure the performance of VQ. Also, the same process was repeated using Gaussian pdf. The test image is replaced and played without and with Gaussian pdf generated code words to observe the effect of VQ.

TSVQ is constrained VQ technique for data compression, where the code vectors form a highly tree structure. To simulate the TSVQ, a training sequence and a test image is required. Table 5.7 shows detailed parameters of both the images. There is no specific criterion for selecting test image. Although it is selected such that variations in the test image can be taken care by the codebook image.i.e.No variations larger than the code book image.

Image file/Properties	**'cameraman.tif'**	**'moon.tif'**
Use	Codebook Generation	Test Image
Resolution	256 x 256	537 x 358
Pixel	8-bit (uint 8array)	8-bit (uint 8array)
Total Samples	65536	192246
Format	tif	tif
Image Type	Monochrome	Monochrome

Table 5.7 Images used for TSVQ Simulation

For VQ=1 i.e.SQ, at the root of the tree is single centroid of length 1 and size 1.To generate *node splitting criteria* is applied as it was in the case of speech simulations. At each root and every sub-root two branches are formed. Є=0.01* 255 is used as the splitting factor to traverse the tree. At level 1 of the tree, from centroid's value, value of Є is added in the right side and value of Є is subtracted in the left side. This process is continued till last tree depth is reached. At level 1 of the tree, the size of codebook is two. At level 2 of the tree, the size of codebook is two. At level 3 of the tree, the size of codebook is three and so on.

The performance of a compressed image using TSVQ is also measured with Peak Signal to Noise Ratio (PSNR). The PSNR defined by normalizing the signal power by the quantization error power and taking a scaled logarithm:

$$\mathbf{PSNR = 10 \log_{10} ((2^n - 1)^2 / MSE)} \qquad (5.3)$$

measured in dB, where n is the number of bits per symbol , MSE is Mean Squared Quantization Error.As an example, if we want to find the PSNR between two 256 gray level images, then we set n to 8 bits. It is usually assumed that the input process has zero mean and hence that the signal power is same as the variance σ^2 of the signal samples. Using images as stated in Table 5.7, we obtained the result of PSNR vs Tree depth (L) for different VQ dimension without Gaussian pdf.

The figure 5.6 depicts resultant plot for VQ dimensions 1 to10 and tree depth 1 to 10.

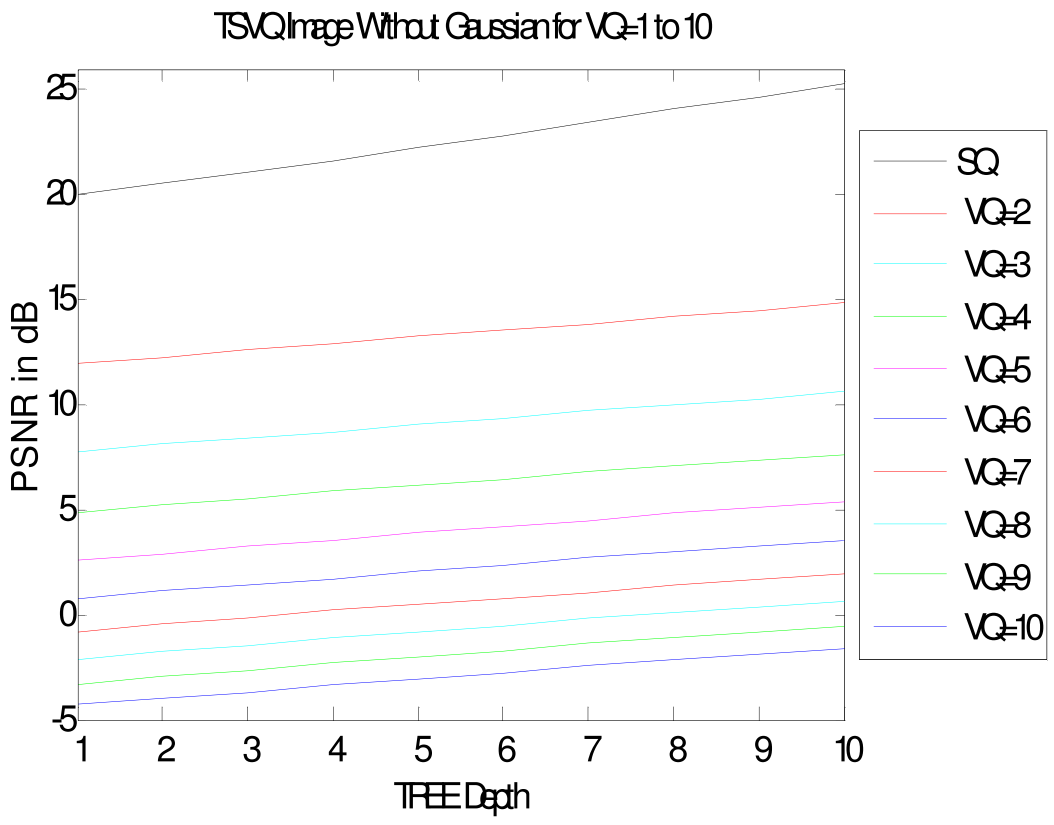


Figure 5.6. SQNR versus Tree Depth for TSVQ Encoded Image with Uniform pdf for VQ=1 to 10

Table 5.8 shows values of PSNR in dB for image compression without Gaussian pdf.

Tree Depth (*L*)	**VQ Dimension**									
	1(SQ)	2	3	4	5	6	7	8	9	10
1	19.987	11.917	7.7436	4.8690	2.590	0.7679	-0.796	-2.105	-3.295	-4.339
2	20.529	12.243	8.0693	5.1919	2.911	1.0905	-0.477	-1.783	-2.973	-4.017
3	21.082	12.569	8.3931	5.5125	3.230	1.4106	-0.161	-1.464	-2.654	-3.698
4	21.647	12.893	8.7147	5.8304	3.546	1.7274	0.1519	-1.148	-2.339	-3.382
5	22.224	13.217	9.0338	6.1451	3.859	2.0404	0.4613	-0.836	-2.028	-3.072
6	22.813	13.539	9.3497	6.4561	4.167	2.3490	0.7664	-0.529	-1.721	-2.765
7	23.416	13.859	9.6622	6.7629	4.469	2.6524	1.0660	-0.227	-1.420	-2.464
8	24.033	14.178	9.9706	7.0649	4.766	2.9500	1.3609	0.0687	-1.125	-2.169
9	24.664	14.496	10.275	7.3615	5.057	3.2408	1.6489	0.3575	-0.838	-1.882
10	25.311	14.812	10.573	7.6521	5.340	3.5242	1.9299	0.6381	-0.558	-1.604

Table 5.8 PSNR (dB) for TSVQ Encoded Image with Uniform pdf.

Using Images as stated in Table 5.7, we obtained the result, PSNR vs Tree depth (*L*) for different VQ dimension with Gaussian pdf. Table 5.9 shows the values of PSNR in dB for image compression with Gaussian pdf.

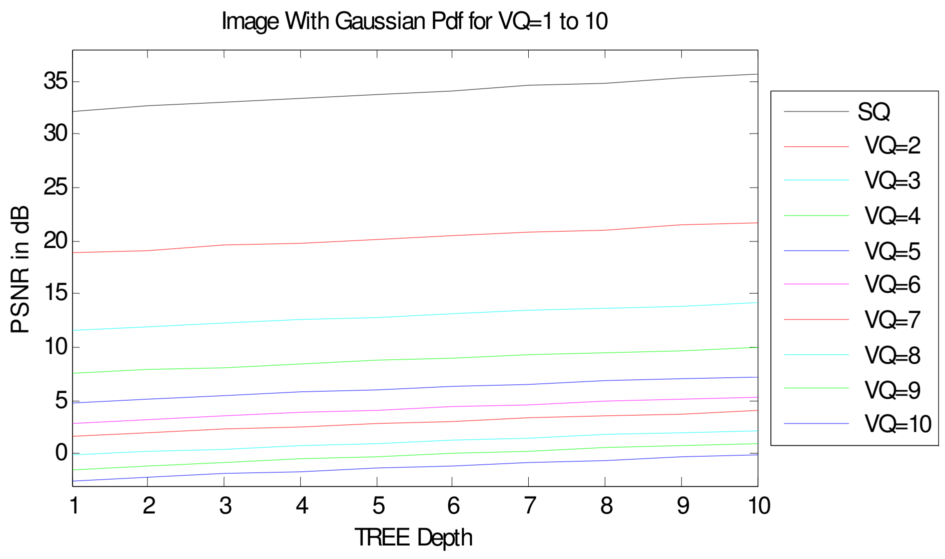


Figure 5.7. SQNR Versus Tree Depth for TSVQ Encoded Image With Gaussian pdf for VQ=1 to 10

Tree Depth (L)	***VQ Dimension***									
	1(SQ)	2	3	4	5	6	7	8	9	10
1	32.226	18.873	11.598	7.5007	4.8438	2.9165	1.6716	-0.1649	-1.4384	-2.5212
2	32.596	19.150	11.915	7.8269	5.1546	3.2270	1.9805	0.14046	-1.1345	-2.2180
3	33.075	19.549	12.251	8.1369	5.4565	3.5292	2.2791	0.43811	-0.8380	-1.9222
4	33.351	19.806	12.542	8.4342	5.7483	3.8219	2.5658	0.72695	-0.5501	-1.6349
5	33.805	20.198	12.856	8.7206	6.0287	4.1038	2.8395	1.0059	-0.2717	-1.3570
6	34.075	20.443	13.124	8.9925	6.2966	4.3738	3.0995	1.2738	-0.0040	-1.0898
7	34.538	20.828	13.415	9.2515	6.5512	4.6308	3.3458	1.5295	0.2519	-0.8342
8	34.810	21.062	13.657	9.4940	6.7914	4.8735	3.5780	1.7718	0.4949	-0.5915
9	35.285	21.439	13.923	9.7220	7.0163	5.1008	3.7970	1.9996	0.7238	-0.3628
10	35.564	21.658	14.138	9.9326	7.2253	5.3118	4.0031	2.2116	0.9375	-0.1493

Table 5.9 PSNR (dB) for TSVQ Encoded Image with Gaussian pdf.

To check the improvement in performance, PSNR using Gaussian pdf, figure 5.8 shows resultant plot for PSNR vs Tree Depth for VQ dimensions of 6 and Tree depth 1 to 10.

Although the PSNRs of the reconstructed images that are decoded with Gaussian Pdf compression method and that with Gaussian pdf compression method are seems to be improved very much.

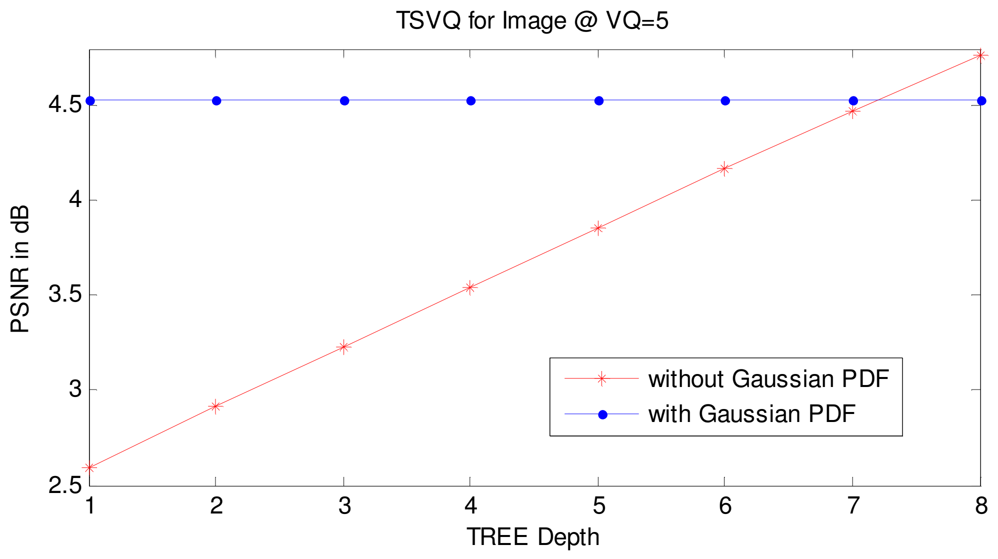


Figure 5.8. PSNR Versus Tree Depth for TSVQ Encoded Image With Uniform pdf and With Gaussian pdf for VQ=5.

Using images as stated in Table 5.7, we retrieve the resultant image for different VQ dimension without Gaussian pdf. Figure 5.9 shows the resultant reconstructed image without Gaussian pdf for VQ=2, Tree Level= 10.

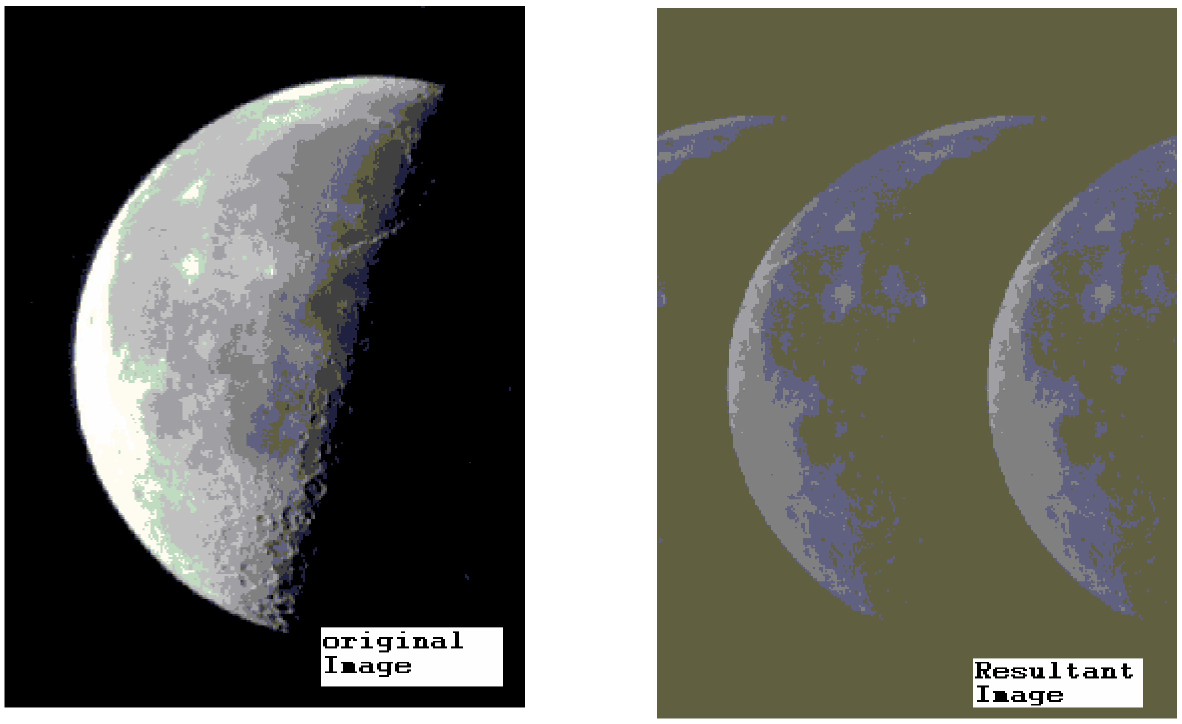


Figure 5.9. TSVQ decoded Image with Uniform pdf for VQ=2 and tree level =10.

Using images as stated in Table 5.7, we retrieval the resultant images for different VQ dimension with Gaussian pdf. Figure 5.10 shows resultant the reconstructed image with Gaussian pdf for VQ=2, Tree Level=10.

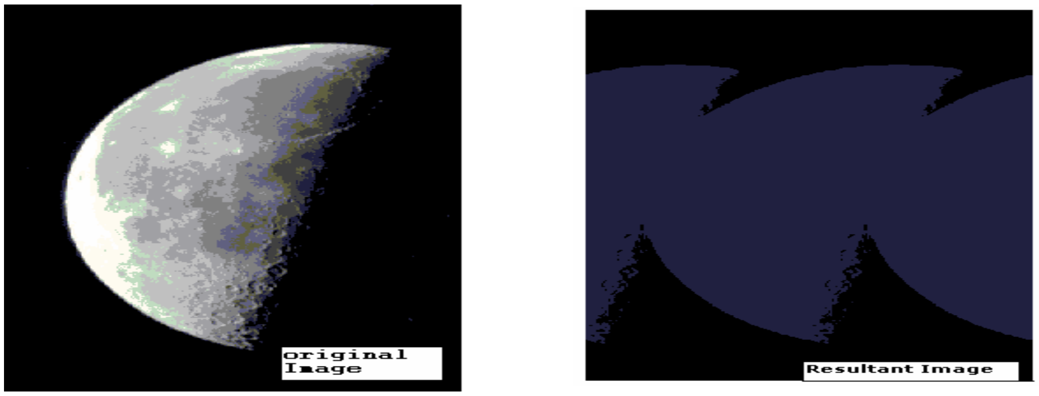


Figure 5.10 TSVQ decoded Image with Gaussian pdf for VQ=2 and tree level =10.

Bits per pixel

A natural way to apply TSVQ to images is to decompose a sampled image into rectangular blocks of fixed size and use this blocks as the vector. For example, each vector may consist of a square block of 2 x 2 of picture elements or *pixels*, so each vector has 4 coordinates. A digital image has a resolution of 8 bits per pixel (bpp). The goal of VQ is to reduce this less than 1 bit per pixel without perceptible loss of picture quality. The table 5.15 indicates bpp value for SQ as well as VQ =2 to VQ=10.

VQ Dimension	Image block Size	Bits per pixel
1(SQ)	1 x 1	1
2	2 x 2	0.5
3	3 x 3	0.33
4	4 x 4	0.25
5	5 x 5	0.20
6	6 x 6	0.167
7	7 x 7	0.143
8	8 x 8	0.125
9	9 x 9	0.111
10	10 x 10	0.1

Table 5.10 Bits per pixel and VQ Dimension for TSVQ

Figure 5.11 depicts PSNR vs bits per pixel (bpp) for tree depth of 5.

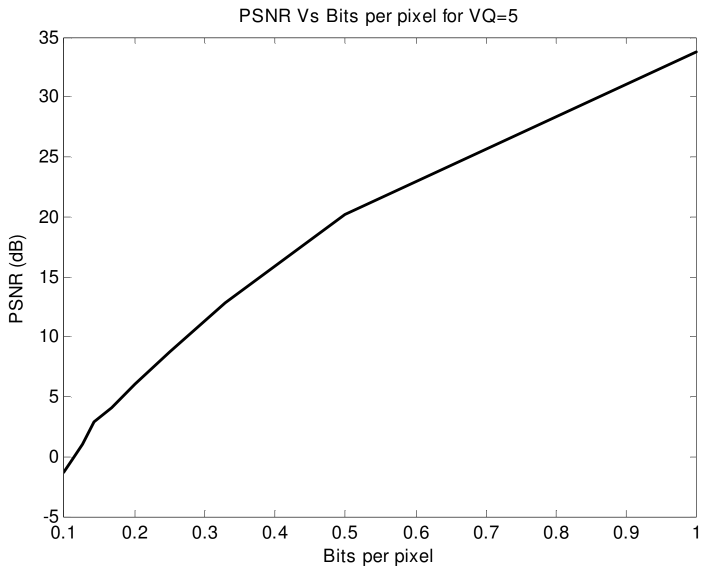


Figure 5.11 PSNR vs bits per pixel (bpp) for tree depth of 5.

We can observe that the Tree-structured vector quantization (TSVQ) is a highly efficient technique for locating an appropriate codeword for each input vector. The algorithm does not guarantee that the selected codeword is the closest one to the input vector. Consequently, the image quality of TSVQ is worse than that of full-search VQ (FSVQ).The evaluation criteria for the comparison is SQNR. FSVQ with 256 code words applied on the image yields -20.241 and -23.372 dB for Uniform and Gaussian pdf respectively for 4x4 blocks (Refer to Table 4.8). While for the same number of code words, 256, with tree depth = 8, TSVQ applied on image yields -33.952 and -27.069 dB respectively for Uniform and Gaussian pdf respectively (Refer to Table 5.8 and 5.9).Here PSNR is normalized by subtracting $20\log_{10}(255)$ = 48.131 dB in the respective observation values of Table 5.8 and 5.9. Comparing these values we find that SQNR performance in TSVQ is worse than that of FSVQ by 13.711 and 3.697 dB for Uniform and Gaussian pdf respectively.

5.4 Summary

This chapter explains the TSVQ approach for compression in general.This includes the operation of TSVQ encoder. Tree generation approach with node splitting criteria, consideration of the codebook size, computational and memory requirements and tree-searching algorithm is discussed in detail. Technique of design and simulations of TSVQ for speech signal is applied. Extensive computer simulations were carried out to evaluate the performance of design. Simulation used two uncoded speech signal, one to generate tree structured codebook and another speech signal is used as test file. SQNR is obtained for VQ dimensions 1 to 10 with tree depth of 1 to 10 for without Gaussian pdf and with Gaussian pdf. For reconstructed signal VQ dimension and tree depth are to be decided by the user.

We also described method of design and simulations of TSVQ for Image. Extensive computer simulations were carried out to evaluate the performance of design. TSVQ is applied to speech for VQ dimensions 1 to 10 and tree depth 1 to 10 with Uniform as well as Gaussian pdf. Similarly TSVQ is applied to image for VQ dimensions 1 to 10 and tree depth 1 to 10 with Uniform as well as Gaussian pdf. In Gaussian pdf better performance is obtained compared to Uniform pdf in both the cases. Simulation code used two monochrome images, one to generate tree structured codebook and another image is used as a test file. PSNR is obtained for VQ dimensions 1 to 10 with tree depth of 1 to 10 for without Gaussian pdf and with Gaussian pdf. For the reconstructed image VQ dimension and tree depth are to be decided by the user.

Chapter 6

Multi Stage Vector Quantizer

6.1 Multi Stage Vector Quantization

Multi Stage Vector Quantization (MSVQ) is a very efficient signal coding technique. It has met with considerable success in audio coding, and has recently been found to be applicable also to image and video coding (71). The low complexity and memory requirements were among the most attractive features of MSVQ. In the case of images, blocks of uniform size are input to the quantizer which in turn finds the best set of MSVQ-parameters.MSVQ decomposes the source vector into the sum of Code vectors, one per stage. Historically, MSVQ was conceived as a sequential quantization operation where each stage simply quantizes the residual of the previous stage. More recently, the greedy nature of simple sequential encoding was recognized, and efficient techniques were proposed to seek better approximation of the source vector as combination of stage-vectors (72).

In all prior work, it was implicitly assumed that scalability is accomplished by using a tree-structured decoding scheme, i.e., reproduction at layer is an unconstrained function of all encoding indices from layer to layer. Of course, tree structured decoding is not a special case, but rather, it is the most general, and hence the optimal scalable coding strategy. However, its natural implementation, tree-structured vector quantization (TSVQ) (73), is usually impractical due to its high codebook storage requirements and demand for a huge training set in the design stage. To mitigate these complexity barriers, in most practical applications such as speech coding, a special case of TSVQ, namely, the multistage vector quantization (MSVQ) (73), is preferred. MSVQ is what we call an “additive” refinement structure, because refinement is based on adding a new vector, which is a function of only the current layer encoding index, to the previous layer reconstruction. In Figure 6.1, the decoding structures of TSVQ and MSVQ are compared.

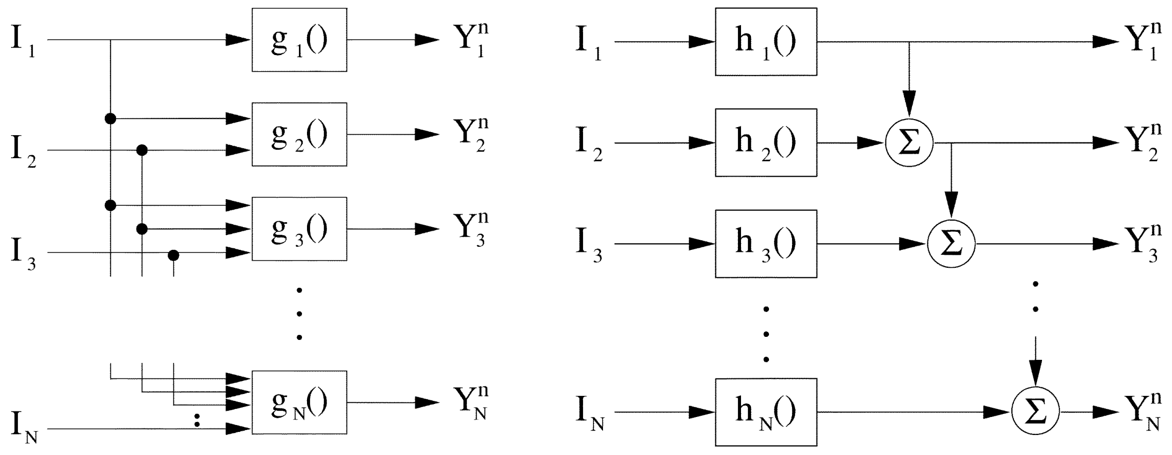


Figure :- 6.1 The TSVQ (left) and MSVQ (right) decoders

An M-stage MSVQ consists of M codebooks $C_1, C_2 \ldots C_M$. The codebook C_i is a set of 2^{ri} code vectors addressable by a r_i bit index I_i . A given source vector **x** is approximated by

$$\mathbf{x}^{\wedge} = u_1(I_1) + u_2(I_2) + \ldots + u_M(I_M) \quad (6.1)$$

where $u_i(I_i)$ is a code vector in Ci , which is indexed by I_i .The objective of the encoding operation is to select a code vector from each codebook such that the error d(**x, x**$^{\wedge}$) is minimized where d(.,.) is a distortion measure. The set of indices ($I_1, I_2, \ldots, I_M$) is transmitted and allows decoder to produce **x**$^{\wedge}$.

If a random vector **x** is such that it does not have a wide variation of gain or of mean values, then shape-gain or mean-removed VQ methods are not likely to be very helpful. In mean-removed VQ mean is removed from the vector and VQ is performed over mean-removed residual. In shape – gain VQ root mean-square value of the vector components (gain) and normalized input vector (shape) are vector quantized. If the dimension is quite large, partitioned VQ would certainly solve the complexity problem but might severely degrade performance when there is substantial statistical interdependence between different sub vectors. If we are as concerned with storage as with search complexity, then general tree-structured VQ and classified VQ are not helpful.

Furthermore, transform VQ may be of very limited help if the degree of compaction achievable still results in too high a vector dimension.

One alternative technique that has proved valuable in a number of speech and image coding applications is multistage or cascaded VQ (74).The technique is also sometimes referred to as residual VQ (37) (75).The basic idea of multistage VQ (MSVQ) is to divide the encoding task into successive stages, where the first stage performs a relatively crude quantization of the input vector using a small codebook. Then, a second stage quantizer operates on the error vector between the original and quantized first stage output. The quantized error vector then provides a second approximation to the original input vector thereby leading to a refined or more accurate representation of the input. A third stage quantizer may then be used to quantize the second stage error vector to provide a further refinement and so on. In comparison with a Full Search single quantizer with same total number of bits and operating on GLA, a two-stage quantizer has the advantage that the codebook size of each stage is considerably reduced so that the total search complexity is substantially lowered. The price paid for this advantage is an inevitable reduction in overall SNR achieved with two stages. This has been shown by Gersho and Gray (73).

We consider first the special case of two-stage VQ as illustrated in figure 6.2.The input vector **X** is quantized by the initial or first stage vector quantizer denoted by Q_1.The quantized approximation $\mathbf{X}^{\wedge}_1$ is then subtracted from **X** producing error vector E_2.This error vector is then applied to a second vector quantizer Q_2 yielding the quantized output $\mathbf{E}^{\wedge}_{\mathbf{2}}$.The overall approximation $\mathbf{X}^{\wedge}$ to the input **X** is formed by summing the first and second approximations, $\mathbf{X}^{\wedge}_{\mathbf{1}}$ and $\mathbf{E}^{\wedge}_{\mathbf{2}}$. The encoder for this VQ scheme simply transmits a pair of indexes specifying the selected code vectors for each stage and the task of the decoder is to perform two table lookups to generate and then sum the two code vectors.

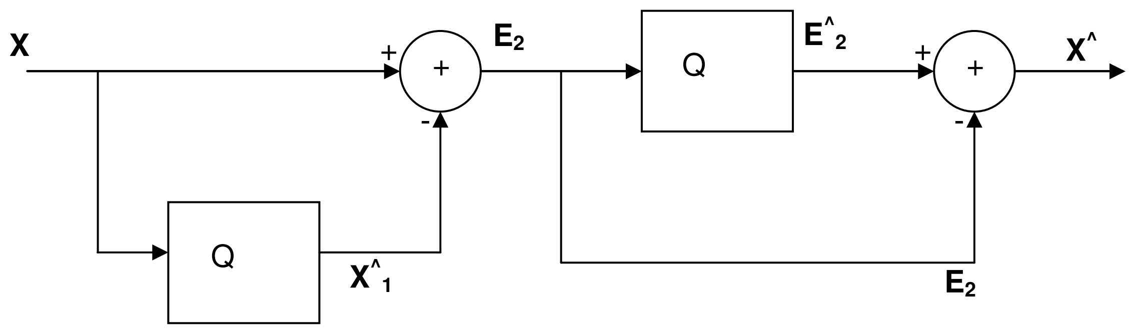


Figure: - 6.2 Two-Stage VQ

By inspection of the figure it may be seen that the input-output error is equal to the quantization error introduced by second stage, i.e. $\mathbf{X} - \mathbf{X}^{\wedge} = \mathbf{E_2} - \mathbf{E}^{\wedge}_{\mathbf{2}}$. From this equation it can be readily seen that the signal to quantization noise power ration in dB (SNR) for the two stage quantizer is given by SNR = SNR1 + SNR2, where SNR_i is signal–to–noise ratio in dB for the i^{th} quantizer. In comparison with a single quantizer with same total number of bits, a two-stage quantizer has the advantage that the codebook size of each stage is considerably reduced so that the total search complexity is substantially lowered. The price paid for this advantage is an inevitable reduction in overall SNR achieved with two stages.

The two stages VQ scheme can be viewed as two level TSVQ.To see this, let $K_1 = \{X^{\wedge}_{1,j}; j=1\ldots N1\}$. Denote the codebook of quantizer Q_1 and $K_2 = \{ \mathbf{X}^{\wedge}_{2,j} ; j=1,\ldots,N_2\}$. Denote the codebook of quantizer Q_2. Then the first level TSVQ codebook coincides with the first stage codebook, i.e. $C^* = K_1$. The second level TSVQ codebooks correspond to shifted versions of the second stage codebook K_2,i.e., $C_j = \{ \mathbf{X}^{\wedge}_{1,j} + \mathbf{E}^{\wedge}_{2,j} ; i =1,\ldots, N_2 \}$. Thus the 2-stage VQ structure is exactly equivalent to specific two-level codebooks.

The general multistage VQ method can be generated by induction from two-stage scheme. By replacing the box labeled Q_2 in figure 6.2, with a two-stage VQ structure, we obtain 3-stage VQ. By replacing the last stage of an m-stage structure, we increase the number of stages to m+1. The general configuration for an MSVQ encoder is easily inferred from figure 6.3 which illustrates 3 stage case. The corresponding decoder is shown in figure 6.4.

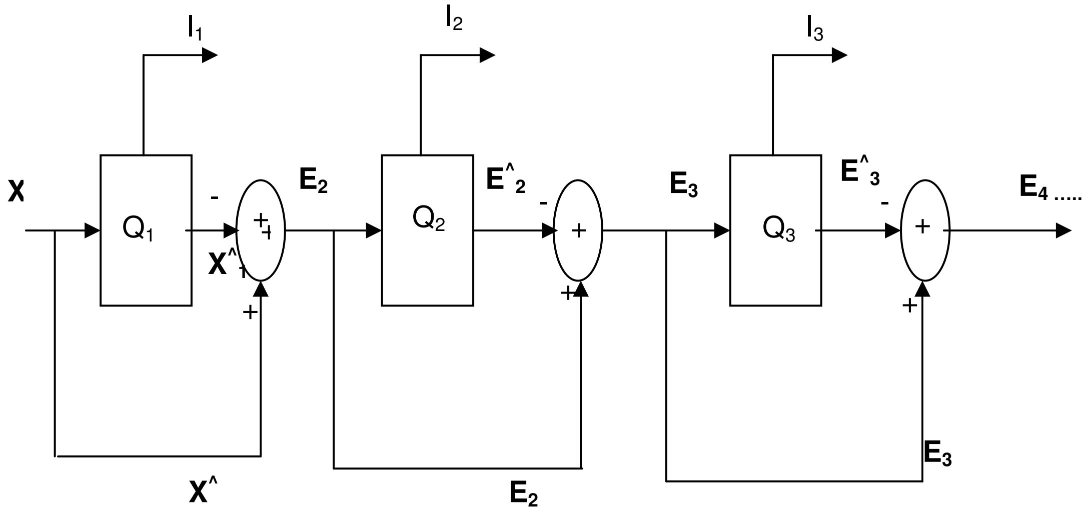


Figure :- 6.3 Multistage Encoder

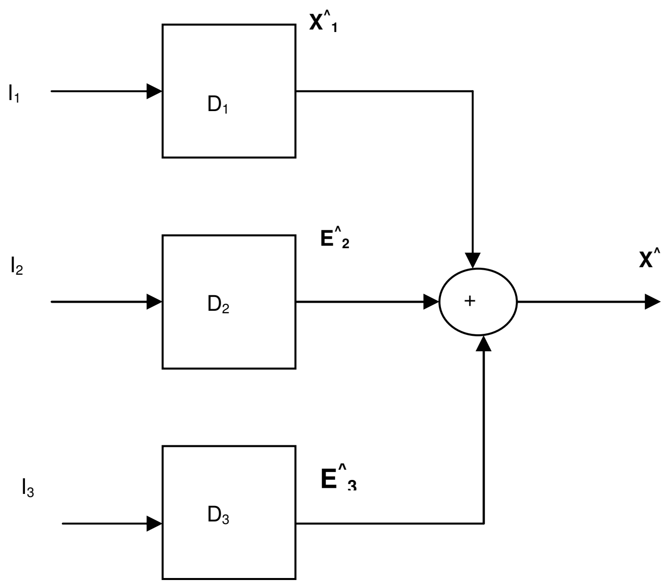


Figure :- 6.4 Multistage VQ Decoder

The vector quantizer at each stage operates on the quantization error vector from the previous stage. Each stage generates an index that is sent to the decoder.

In the multistage approach , the input vector is represented as the sum of two or more vectors of the same dimension as the original, where each successive term of the sum can be considered as a refinement or successive approximation improvement of previous terms. The reproduction vector is computed in the following form:

$$\mathbf{X}^{\wedge} = \mathbf{X}^{\wedge}_{1} + \mathbf{E}^{\wedge}_{2} + \ldots + \mathbf{E}^{\wedge}_{m} \qquad (6.2)$$

The first stage quantizer approximates **X** with $\mathbf{X}^{\wedge}_{1}$ using a codebook K_1 of size N_1. The second stage quantizer Q_2 is used to approximate the quantization error vector $\mathbf{E_2} = \mathbf{X} - \mathbf{X}^{\wedge}_{1}$ using a codebook K2 so that

$$\mathbf{X}^{\wedge}_{2} = \mathbf{X}^{\wedge}_{1} + \mathbf{E}^{\wedge}_{2} \qquad (6.3)$$

is an improved approximation to **X**. Similarly, the third stage quantizer approximates the quantization error vector $\mathbf{E_3} = \mathbf{X} - \mathbf{X}^{\wedge}_{1} - \mathbf{E}^{\wedge}_{2}$ associated with the second stage using the codebook K_3 to obtain

$$\mathbf{X}^{\wedge}_{3} = \mathbf{X}^{\wedge}_{1} + \mathbf{E}^{\wedge}_{2} + \mathbf{E}^{\wedge}_{3} \qquad (6.4)$$

This provides a further refinement to **X**; and so on, until the complete approximation is produced by adding the m vectors. There is a separate codebook K_i for each of the m vectors that form the reproduction vector. The overall codeword is the concatenation of code words or indices chosen from each of these codebooks. The encoder transmits indexes I_1, I_2, …, I_m to the decoder, which then performs a table-lookup in the respective codebooks and forms the sum .Thus this is a product code where the composition function g of the decoder is simply a summation of the reproductions from the different VQ decoders.

Note that the complexity is reduced from $N = \prod^{m}_{i=1} N_i$ to $\sum^{m}_{i=1} N_i$ and the equivalent product codebook is generated from the Cartesian product $K_1 \times K_2 \times \ldots \times K_m$. Thus both the complexity and storage requirements can be greatly reduced by using multistage VQ.

As usual, there is a performance penalty with this product code technique. It is easy to show that the overall quantization error between input and output is equal to the quantization error introduced in the last stage; from this it can be shown that the signal to quantizing noise power ratio in dB (SNR) for the multistage quantizer is given by

$$SNR = \sum^{m}_{i=1} SNR_i \quad (6.5)$$

Consistent with the two-stage result discussed earlier. If we make the idealized assumption that the SNR of a quantizer grows linearly with the number of bits, this result would indicate that we can achieve the same performance by sharing a fixed quota of bits over several quantizers as by using all the bits in one quantizer while at the same time achieving a greater complexity reduction. Also, this is not so. The coding gain of a quantizer depends on the statistics of the input signal and the successive quantizers tend to have rapidly diminishing coding gains (i.e. the SNR per bit decreases) due to the vector components of successive stages tending to be less correlated. Generally the quantization error vector has more randomness than the input to the quantizer since its components tend to be less statistically dependent than those of the input vector. Thus, in practice, multistage coders often have only two and occasionally three stages. As far as we know, there has been no report of coding system using four or more stages.

6.2 Design of Multi Stage VQ

Codebook design for multistage VQ is also performed in stages. First, the original training set T is used to generate the first stage codebook of the desired size in conventional manner. Next, a new training set T_1 is generated by applying the training set to the first stage VQ and generating the set of error vectors that represent the statistics of the vectors applied to the second stage. This training set is of the same size as the original and the vectors are of the same dimension. The process is repeated for successive stages. Note that the codebook design complexity is reduced compared to the design for a single stage VQ of dimension N.

This codebook design procedure is not optimal in the sense that it does not find the best set of codebooks for sequentially encoding a vector stage-by-stage with the multistage structure. It is however, greedy in the sense it finds the codebook for the first stage that would be optimal if there were only one stage. Then it finds the best codebook for the second stage given the first stage codebook and assuming there are only two stages. Similarly, each successive stage is optimal given all previous stage codebooks and assuming that it is the last stage. Thus design algorithm is consistent with the philosophy of progressive reconstruction as discussed in the context of TSVQ.

An improved design algorithm for MSVQ was proposed by W.Y. chan , S. Gupta and A. Gresho, which yields slightly better results than the greedy algorithm usually used for the usual sequential –search MSVQ encoder. Alternatively, if the sequential-search encoder is abandoned and exhaustive search is performed to find the best set of indexes { I_1, I_2, …, I_M} for the given m stage MSVQ decoder , even better performance can be achieved for a given set of codebooks. The design of optimal codebooks for this purpose was considered in (37). In this case the search complexity is essentially the same as with a single unstructured VQ codebook; only the storage requirement is reduced. Multistage VQ can not be considered as recursively indexed VQ because encoder generates index of every stage for suitable code word and decoder performs the summation of the code words based on indexes received. So process of MSVQ is not similar to the recursively indexed VQ.

MSVQ can be regarded as a special case of TSVQ. In tree-structured VQ , the number of codebooks grow exponentially with the number of stages (or layers) , whereas in MSVQ the growth is linear with the number of stages.

6.3 MSVQ Results

In the following section MSVQ empirical data (training set) has been applied to generate codebook. Codebook for uniform and Gaussian pdf is generated .The MSVQ is applied to speech signal. There are no specific criteria for selecting the test sequences. The selected test sequences have different sampling frequency and bit /sample

Wave file Properties	**Om.wav**	**Start.wav**	**Type.wav**
Format	wav	wav	wav
Use	Test sequence	Test sequence	Test sequence
Sampling frequency	8kHz	22kHz	11kHz
Bits / sample	8-bit	16-bit	8-bit
Format	PCM	PCM	PCM
Wave Type	Mono	Mono	Mono

Table 6.1. Wave files on which MSVQ applied

SQNR(dB) Versus VQ dimensions for Om.wav are shown in Table 6.2.Number of codeword in first stage and second stage are 4 and 16 respectively. SQNR(dB) Versus VQ Dimensions for MSVQ Encoded Om.wav is in figure 6.5 .

VQ dimension	**SQNR (dB)** Uniform pdf		**SQNR (dB)** Gaussian pdf	
	Stage 1	Stage 2	Stage 1	Stage 2
1	17.225	36.311	16.389	35.424
2	2.6246	7.9435	15.416	31.546
3	3.1239	10.572	13.959	28.576
4	4.3846	11.979	13.249	26.861
5	4.143	10.747	12.519	25.325
6	4.7695	12.28	12.144	24.509
7	6.6824	13.434	11.722	23.539
8	5.6978	12.423	11.404	22.87
9	6.0617	13.146	11.061	22.22
10	6.2706	12.712	10.837	21.752

Table 6.2. SQNR (dB) Versus VQ dimensions for MSVQ Encoded Om.wav

Table 6.2 indicates that MSVQ applied to speech offers better SQNR with Gaussian pdf compared to Uniform pdf for any vector dimension. SQNR in stage 2 is better compared to stage 1 as the number of code words in stage 2 is more compared to number of code words in stage 1.

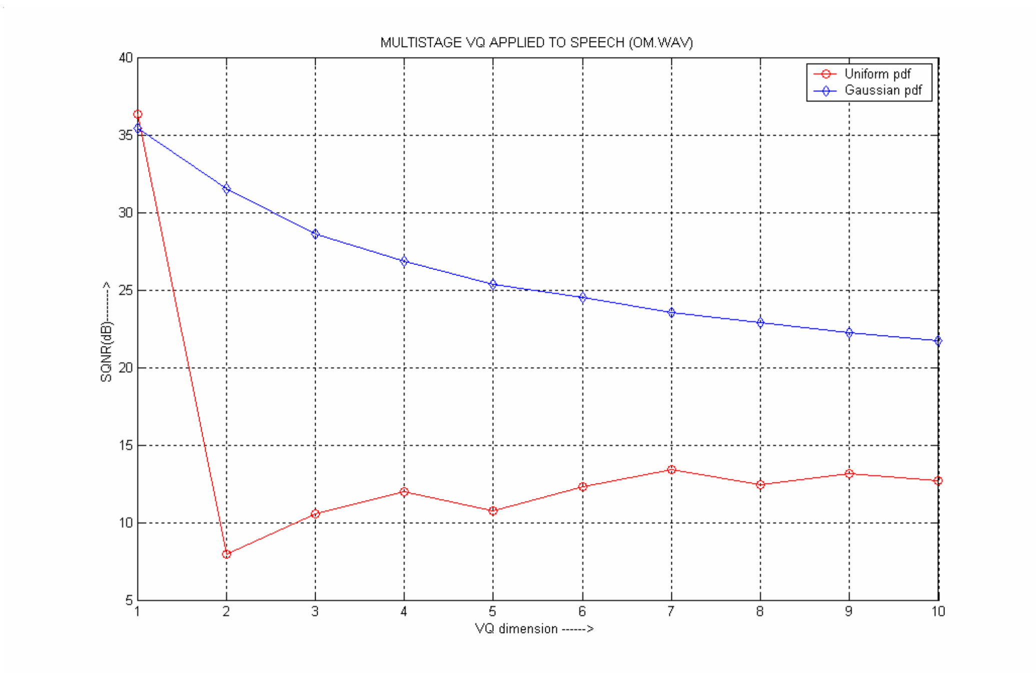


Figure :- 6.5 SQNR(dB) Versus VQ Dimensions for MSVQ Encoded Om.wav

Figure 6.5 indicates better performance of speech encoded with Gaussian pdf compared to speech encoded with Uniform pdf.

SQNR(dB) Versus VQ dimensions for Start.wav are shown in Table 6.3 Number of codeword in first stage and second stage are 4 and 16 respectively. SQNR(dB) Versus VQ Dimensions for MSVQ Encoded Start.wav is in figure 6.6

VQ dimension	**SQNR (dB)** Uniform pdf		**SQNR (dB)** Gaussian pdf	
	Stage 1	Stage 2	Stage 1	Stage 2
1	8.9726	23.857	14.855	33.534
2	3.922	13.634	13.013	26.501
3	4.9764	13.167	11.504	23.289
4	5.4603	11.905	10.729	21.705
5	5.3476	11.161	10.142	20.47
6	4.5993	10.904	9.7143	19.521
7	5.292	11.173	9.3264	18.733
8	6.6898	13.826	8.9934	18.063
9	4.9022	11.738	8.7121	17.471
10	5.9755	12.396	8.4658	16.981

Table 6.3. SQNR (dB) Versus VQ dimensions for Start.wav

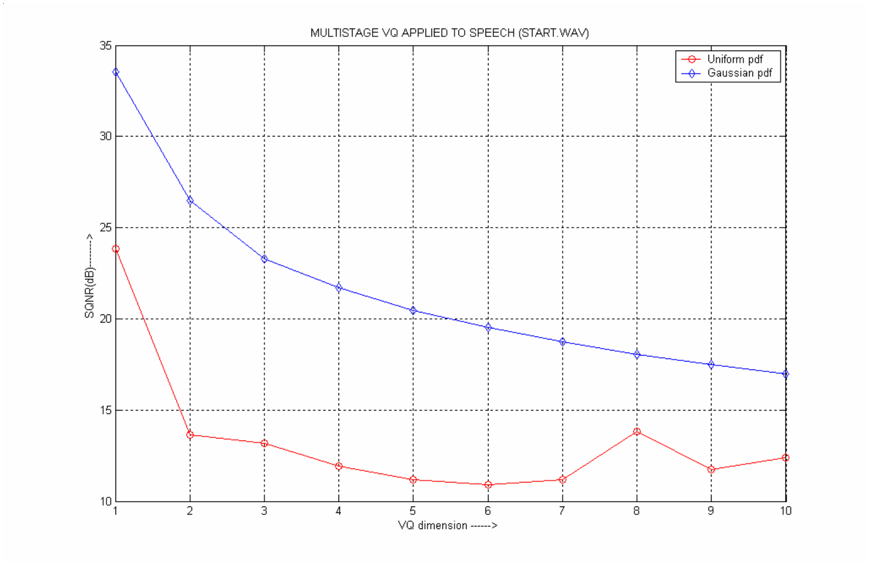


Figure :- 6.6 SQNR(dB) Versus VQ Dimensions for MSVQ Encoded Start.wav

Figure 6.6 indicates better performance of speech encoded with Gaussian pdf compared to speech encoded with Uniform pdf.

SQNR(dB) Versus VQ dimensions for Type.wav are shown in Table 6.4 Number of codeword in first stage and second stage are 4 and 16 respectively. SQNR(dB) Versus VQ Dimensions for MSVQ Encoded Type.wav is in figure 6.7

VQ dimension	**SQNR (dB)** Uniform pdf		**SQNR (dB)** Gaussian pdf	
	Stage 1	Stage 2	Stage 1	Stage 2
1	8.9167	23.449	10.255	23.487
2	4.0914	12.521	8.512	17.253
3	4.8514	11.96	7.1799	14.476
4	4.7202	10.337	6.3935	12.97
5	4.2608	9.2944	5.8208	11.739
6	3.8169	8.9296	5.3713	10.794
7	3.8364	8.5428	4.9898	10.03
8	4.1725	9.1282	4.6678	9.3747
9	3.3929	7.9661	4.3703	8.7619
10	3.848	8.2201	4.0967	8.2142

Table 6.4. SQNR (dB) Versus VQ dimensions for Type.wav

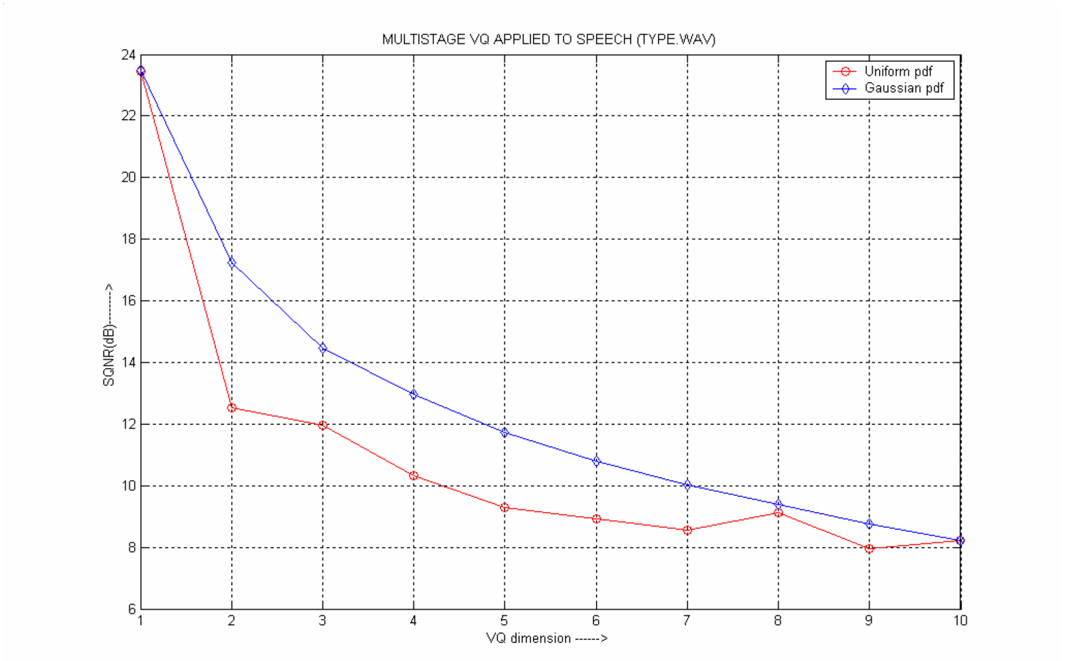


Figure :- 6.7 SQNR(dB) Versus VQ Dimensions for MSVQ Encoded Type.wav

Figure 6.7 indicates better performance of speech encoded with Gaussian pdf compared to speech encoded with Uniform pdf. Comparing the results on the above three different speech , typically 9.074 , 4.237 and 0.247 dB better SQNR is obtained with Gaussian pdf compared to Uniform pdf.

Multi stage VQ is applied to a 256x256 pixels test image , cameraman.tif ,as shown in Figure 4.6 , with the ic.tif , Figure 4.5 , being a 256x256 pixels training image with two stages. SQNR (dB) results with Uniform and Gaussian pdf for different block size are shown below. Number of code words for stage 1 are 16 and number of code words for stage 2 are 32.

4 x 4 Block		
	Stage 1 **No. of Code words =16**	**Stage 2** **No. of Code words = 32**
SQNR(dB) **(Uniform pdf)**	-21.635	-19.173
SQNR(dB) **(Gaussian pdf)**	-25.784	-24.779

Table 6.5 MSVQ Stage wise SQNR (dB) for test image (4 x 4 block)

8 x 8 Block		
	Stage 1 **No. of Code words =16**	**Stage 2** **No. of Code words = 32**
SQNR(dB) **(Uniform pdf)**	-27.435	-26.081
SQNR(dB) **(Gaussian pdf)**	-29.329	-28.969

Table 6.6 MSVQ Stage wise SQNR (dB) for test image (8 x 8 block)

16 x 16 Block		
	Stage 1 **No. of Code words =16**	**Stage 2** **No. of Code words = 32**
SQNR(dB) **(Uniform pdf)**	-31.852	-31.229
SQNR(dB) **(Gaussian pdf)**	-32.670	-32.502

Table 6.7 MSVQ Stage wise SQNR (dB) for test image (16 x 16 block)

Observing table 6.5, 6.6 and 6.7, as the block size increases SQNR decreases for uniform as well as Gaussian pdf. Stage 2 of MSVQ offers better performance as number of code words in stage 2 are more than number of code words in stage 1.Image quantized with Gaussian pdf is not better than image quantized with Uniform pdf. This indicates that pixels in the image are uniformly distributed.

6.4 Summary

This chapter described the definition and concept of MSVQ. Comparison of MSVQ with TSVQ is made. MSVQ can be considered to be a special case of TSVQ. Design of MSVQ is explained. MSVQ design is applied on speech. SQNR for various VQ dimension are obtained. MSVQ design is also applied to image. In the case of speech better performance is found with Gaussian pdf compared to Uniform pdf. Image quantized with Gaussian pdf is not better than image quantized with Uniform pdf. This indicates that pixels in the image are uniformly distributed.

Chapter 7

Trellis Coded Vector Quantizer

7.1 Trellis Coded Vector Quantizer

Trellis coded VQ (TCVQ) can be viewed as an extension of the Finite State VQ approach with a look-ahead search. In this chapter, the concepts and design algorithm for a class of vector quantization systems with memory that are more general are discussed. Before we discuss on trellis coded VQ, delayed decision encoder is discussed. TCVQ reduces the search complexity required in VQ. The basic idea of TCVQ is to search the small codebooks in subsequent stages.

7.1.1 Delayed Decision Encoder

Consider a recursive vector quantizer, which has a decoder $\hat{x}$=β(u,s), where s is the decoder state and u the received channel index, an encoder u= α(x,s), where x is the input vector and s the encoder state, and a next-state rule *f*(u,s) giving the next code state if the current state is s and the current channel word is u. It is assumed in a recursive quantizer that there are no channel errors and hence if both encoder and decoder begin in a common state, then the decoder state and encoder state are thereafter identical for the encoding and decoding operation.

In an ordinary recursive quantizer, the encoder mapping α is in fact determined by the decoder mapping: α(x,s) must produce the channel symbol u which minimizes the distortion d(x, β(u,s)). Such a algorithm is not inherently optimal because the selection of a low distortion short term path might lead to a bad state and hence higher distortion in the long run. Although a goal in coding system design is to avoid this possibility. One means of improving performance over long run is to permit the encoder to wait longer before producing channel symbols. Instead of comparing a single input vector to a single reproduction and producing the channel word, the encoder could simultaneously consider two consecutive input vectors and compare the resulting distortion with two corresponding reproduction vectors resulting from any two channel codewords. This would allow the encoder to ensure a minimum distortion fit for two consecutive input vectors rather than for just one.

More generally, the encoder for a given recursive VQ decoder could view L successive input vectors, then try driving the decoder with every possible sequence of L channel symbols, and find the sequence of channel symbols that produces the best overall fit, that is, the smallest total distortion between the L input and L reproduction vectors. The encoder can now transmit one or more (up to L) of these channel symbols and a new search is begun.

This is the basic idea behind *delayed decision encoding* and the principle can be applied to any recursive decoder. The approach has been called variously *delayed decision encoding, lookahead encoding, multipath search encoding* and *tree encoding.* As L gets larger this technique guarantees that ever longer sequences of input and reproduction vectors will have the minimum possible distortion for the particular decoder and hence one and consider the encoder to be optimal for the decoder. Obviously, however, the incurred encoding delay can quickly become intolerable in some applications. For example, if the coding is part of a communication link in a feedback control system, the data would be too stale to be useful. On the other hand, in some applications a delay of many vectors is tolerable, e.g., in speech or image processing or one-way digital audio or video broadcasting. Also, the encoding complexity grows exponentially with L and can quickly become prohibitive for high or even moderate rates (in bits per input vector).In fact, there are several variations on this concept. Trellis Coded VQ (TCQ) derives the advantage by exploiting Lattice VQ. In TCQ transition from one state to next state for the next sample is very similar to switching the side in Lattice VQ. This offers quantization in well defined distinct partitions as in Lattice VQ.

7.1.2 Block Delayed Decision Encoder

One way to assure theoretically better long term performance than memory less VQ is to have a delayed decision encoder operate essentially as a block code of “giant” vector quantizer: It begins with an initial state, finds an effective sequence of L channel symbols for L input vectors, and then releases the L channel symbols to the channel. It then proceeds to do the same things for the next group of L input vectors. Typically the scheme will drive the encoder and decoder state back into the initial state s* at the end of one group of L input vectors. By resetting the state prior to encoding each block, there is no memory of prior block used in encoding the current block. Hence, in this case the code behaves like a structurally constrained vector qunatizer of length Lk where k is the input vector dimension. This method offers lower distortion. It looks for minimum distortion over L input vectors rather than single input vector.

7.1.3 Incremental delayed decision encoder

Here the path map is selected as in the block case, but instead of releasing the entire block of L channel words, only one (or some small number *v*) are released and the encoder then again searches ahead L vectors so as to pick a new path and advanced further. Such incremental encoding provides a smoother operations and fewer blocking effects in the data then does encoding an entire L block, but it is not as well understood theoretically and it requires more computation per input vector. All of the *delayed decision encoding* can be used in either block or incremental fashion.

7.1.4 Tree And Trellis Coding

Let the initial state be s* and suppose the channel symbols have an alphabet of size $N = 2^R$, here R is the bit rate. For simplicity we shall consider the code has a rate of one bit per source vector. All of the ideas extend in a straight forward manner with the binary trees and trellises becoming more general N-ary trees and trellises.

To depict the operation of the decoder, we use a directed graph called a tree. The tree is used to represent the time evolution of the decoder's operation so that each path thorough the tree identifies a particular sequence of states taken on by the decoder in successive time instants. Thus, each level of the tree corresponds to the time arrival of a new input vector, whereas in the case of TSVQ tree, the entire tree represents the possible ways in which a single input vector is searched.

The initial state is the root node of the tree and all subsequent nodes will also correspond to decoder states. Suppose the encoder is in the initial state and the channel index is binary valued, either a 0 or 1. If it produces a 0, then the decoder will produce a reproduction $\beta(0,s^*)$ and the decoder will advance to state $s_0= f(0,s^*)$. If the encoder produces a 1, then the decoder will produce a reproduction $\beta(1,s^*)$ and the decoder will advance to state $s_1= f(1,s^*)$. This action is depicted in Figure 7.1 by a branch of the tree moving upward if the encoder produces a 1 and downward if it produces a 0. The branch is labeled by the output produced if that branch is taken and the branch is terminated in a node representing the new state. Thus we have "grown" the first level of a code tree. If the encoder is in state s_0 in level 1, then it can produce a 0 with a resulting output $\beta(0, s_0)$ and the next state $s_{00} = f(0,s_0)$, or a 1 with resulting output $\beta(0, s_1)$ and next state $s_{01} = f(0,s_0)$. For convenience we subscript the states by the binary channel sequence which led from the initial state to the current state. This sequence is called the path map since knowing the sequence one can trace a path through the tree from the root node to the current node. For example, a path map (received sequence) of 1101 will yield a decoded reproduction sequence

$$\beta(1, s^*), \beta(1, s_1), \beta(0, s_{11}), \beta(1, s_{110})$$

Note that with the tree drawn horizontally, the horizontal position of a node corresponds to a particular time, with the root node corresponding to t = 0 and time increasing to the right. Table 7.1 summarizes the actions available to the decoder at the first level of the tree.

While the tree depicts the possible paths of the decoder, it is the encoder which must make use of this tree in order to search for an effective or optimal path and thereby determine the channel symbols to be transmitted. If the encoder is to have a delay or search depth of L input vectors, then the tree should be extended to L levels. Note that the tree grows exponentially fast with the number of levels since each level has twice as many nodes as the previous level. Given such a code tree, the task of the encoder is now the following: Given an input sequence $x_0, x_1, \ldots, x_{L-1}$, find a path map through the tree (a sequence of symbols from the alphabet of size $N = 2^R$) $u_0, u_1, \ldots, u_{L-1}$ such that the corresponding sequence of states $\sigma_0, \sigma_1, \ldots, \sigma_{L-1}$ (where $\sigma_0 = s^*$) and branch labels (outputs) $(\hat{x}_0, \ldots, \hat{x}_{L-1}) = (\beta(\sigma_0, u_0), \ldots, \beta(\sigma_{L-1}, u_{L-1}))$ yields the minimum value of the path distortion

$$\Delta(u_0, u_1, \ldots, u_{L-1}) = \sum_{l=0}^{L-1} d(x_l, \hat{x}_l) \tag{7.1}$$

Where d is the distortion measure between any pair of vectors (x and $\hat{x}$) sometimes called the per-letter distortion.

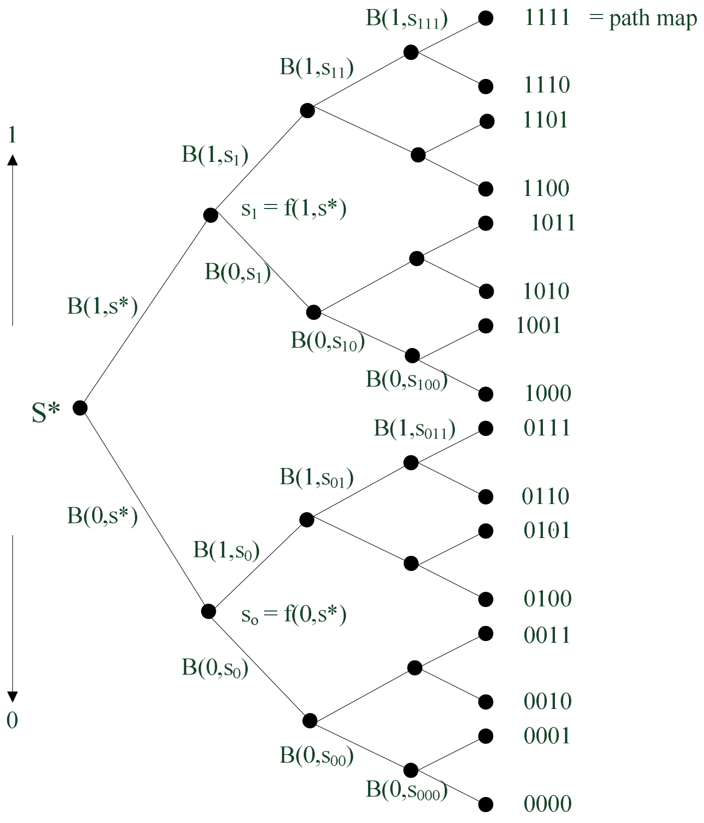


Figure 7.1: Recursive Decoder Tree

Level 1 State	Channel Symbol	Output	Level 2 State
S_0	0	$\beta(0, s_0)$	$s_{00} = f(0,s_0)$
S_0	1	$\beta(1, s_0)$	$s_{01} = f(1,s_0)$
S_1	0	$\beta(0, s_1)$	$s_{10} = f(0,s_1)$
S_1	1	$\beta(1, s_0)$	$s_{11} = f(1,s_1)$

Table 7.1: First Level Decoder Actions

Trellis as a Merged Tree

The recursive VQ decoder is in fact a finite state VQ decoder. With only finite number of states the tree diagram becomes highly redundant. Once one has descended to a sufficient depth of the tree, there are more nodes than there are distinct states of the decoder and hence several of the nodes must represent the same state. The subtree emanating from any node having a given state will be identical for each such node at this level of the tree. No level can have more distinct nodes than there are states. Thus we can simplify the picture if we merge all those nodes corresponding to the same state to produce a simpler picture of the decoding tree, Thus leads to a new and simplified map called a *trellis* that more compactly represents of all possible paths through the tree. If the number of states is K, then we need only consider K nodes, one for each state, at each successive time instant after t = 0. The trellis consists of the set of all paths starting at the root node and traversing one of the K state nodes at each time instant and terminating at a fixed node at time L-1.

The name *trellis* for a merged tree was invented by G.D.Forney, Jr., based on the resemblance of the resulting merged tree to the common garden trellis. A trellis encoding system is a recursive coding system with a finite state decoder and a minimum distortion trellis search encoder. It should be pointed out that the goal of the searches of trees and trellises is to produce a minimum distortion path. The Viterbi algorithm encoder is to be considered shortly indeed finds a minimum distortion path to the search depth of the encoder.

7.1.5 Sliding Block Decoder

A particular type of recursive decoder with a finite number of states, called a *shift register decoder*, consists of a shift register which stores in each stage an R bit binary vector representing a channel symbol and with m stages along with a lookup table having $N^m = 2^{Rm}$ possible reproduction levels as outputs as depicted in Figure 7.2 for the special case of m=3 stages and binary channel symbols, i.e., R=1. The contents of the shift register form an N-ary-tuple $u = (u_0, u_1 \dots u_{m-1})$ (with the smallest index being the most recent entry and hence the leftmost channel symbol in the shift register). If we define the current state to be $s = (u_0, u_1, \dots, u_{m-1})$ then the decoder output mapping is given by $y_u = \beta(u_0,s) = g(u)$. Thus y_u is a code vector from the state codebook Cs and is therefore is an element of the super codebook C of all possible reproduction vectors. The possible outputs of the shift register decoder are contained in the codebook or lookup table C and the N-ary shift register decoder content, u, forms the index that addresses the codeword y_u in the table.

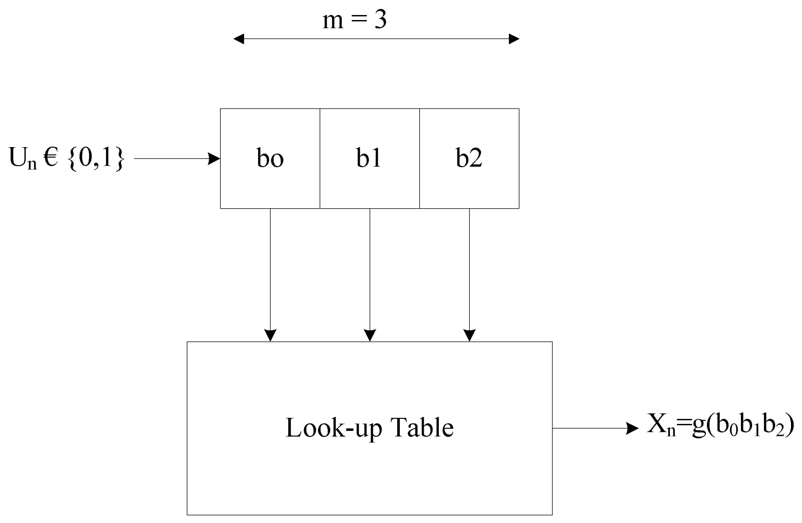


Figure 7.2: Sliding Block Decoder

Such codes are called sliding block codes because a block of input vectors is viewed to produce an output symbol and at successive time instants the new block is formed by "sliding" or shifting the block so that the oldest (rightmost) vector is dropped, a new vector is included at the left, and the remainder are just shifted to the right.

In the example of Figure 7.2, each symbol is one bit and the decoder is a finite-state VQ decoder with four states. The states are the four possible binary pairs constituting the two oldest channel bits in the register. If the shift register content is $u = b_0b_1b_2$, then $u = bs$, where $b = b_0$ is the current channel bit (leftmost bit in the register) and $s = b_1b_2$, the state, comprises the remaining two bits. With this convention the reproduction function becomes simply

$$\beta(u, s) = g(bs) = y_{bs} \qquad (7.2)$$

The next state function is forced by the shift register structure: the next state is always formed by shifting so that the current channel bit becomes the left bit in the state and the previous left state becomes the new right state bit. This is summed up by the next-state table for the given example in Table 7.2. Alternatively it can be summarized by the formula

$$f(b, b_1b_2) = b\, b_1 \qquad (7.3)$$

u	**v**	**$s_{next} = f(u,v)$**
0	00	00
0	01	00
0	10	01
0	11	01
1	00	10
1	01	10
1	10	11
1	11	11

Table 7.2: Sliding Block Next State Function

There is nothing unique about this representation of the decoder as a finite-state decoder, it is also common in the literature for authors to reverse the bits describing the state and stick the channel symbol on the right, that is, to consider the state to be b_2b_1, and to write the output functions as $g(b_2b_1b_0)$ so that the rightmost bit represents the most recent one. This will result in a different but equivalent next state function. For this example we can redraw the decoder tree of Figure 7.1 as the trellis diagram shown in Figure 7.3. The initial state is taken as s* = 00.

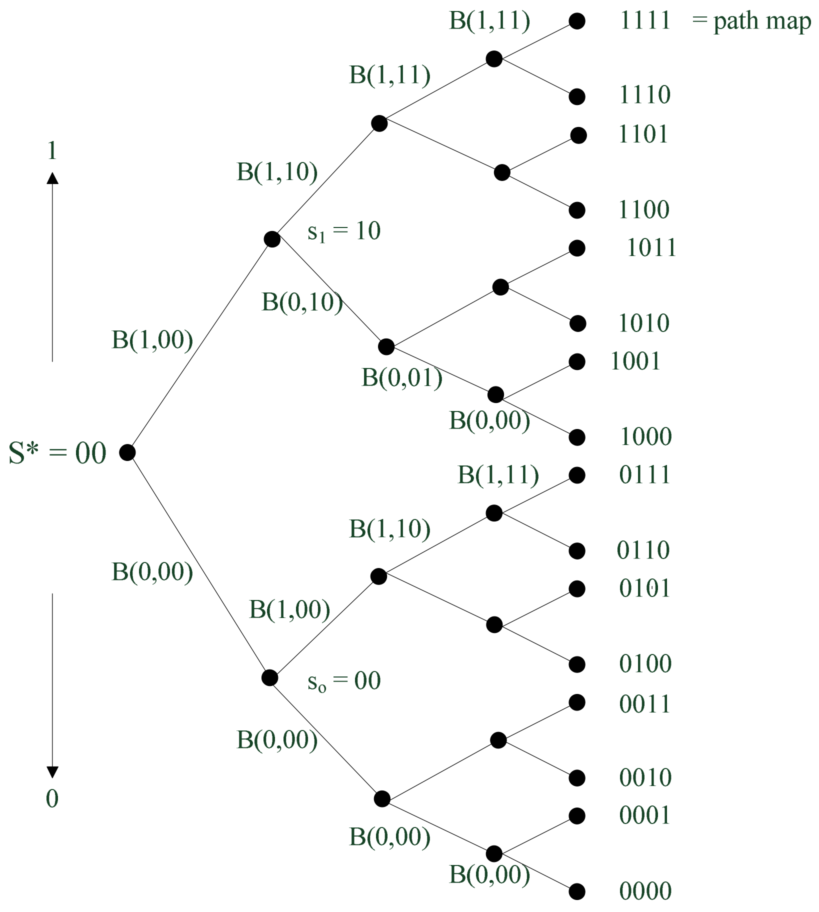


Figure 7.3: Sliding Block Decoder Tree

In memory less VQ it is clear that the larger is k, the better is the rate-distortion tradeoff that can be achieved; of course, complexity places a limit on k for a given rate. On the other hand, in trellis encoding, the performance measure given by per letter distortion shows that the distortion to be minimized depends only on the value of the product kL, the total number of samples under consideration rather than on the block size k. This suggests that there might be no sacrifice in performance by using k = 1, where the vectors are one dimensional. Trellis with four states is drawn in figure 7.4.

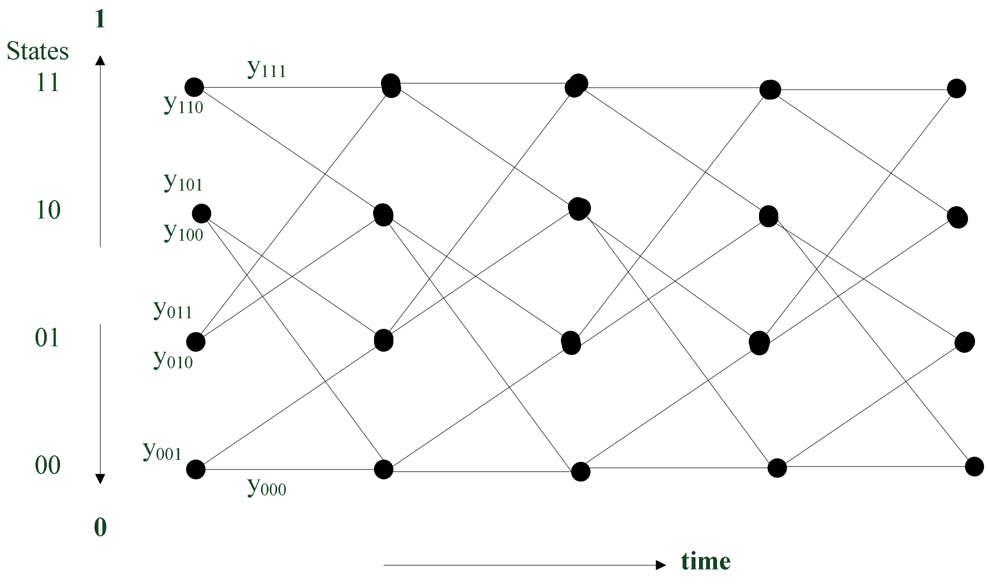


Figure 7.4: Trellis

7.1.6 The Viterbi Algorithm

The Viterbi algorithm is a minimum-cost search technique specifically suited for a trellis. It is an example of dynamic programming and was originally developed for error control codes.

Suppose that we now have a finite-state recursive decoder and hence a decoder trellis as typified in Figure 7.4. The key idea of Viterbi algorithm is following. The minimum distortion path from time 0 to time n must be an extension of one of the minimum distortion paths to a node at time n-1. Thus in order to find the best possible path of length L we compute the best path to each state for each time unit by finding the best extension from the previous states into the current state and we perform this for each time unit up until time L. This idea is known formally as the optimality principle of dynamic programming. The detailed operation is described in Table 7.3.

Viterbi Algorithm Trellis Encoder

1. Given a collection of states $S = \sigma_0, \ldots\ldots, \sigma_k$, a starting state s*, a decoder $\beta(u ,s)$ producing a reproduction given channel symbol u in state s, a per-letter distortion measure d, a source input vector

$x_0, x_1, \ldots, x_{L-1}$. Let $D_j(k)$ denote the distortion for state k at time j. Set $D_0(s^*) = D_0(s) = \infty$ for $s \neq s^*$. Set l = 1.

2. For each current state s find

$$\mathbf{D_l(s) = \min_{\sigma} \left(D_{l-1}(\sigma) + \min_{u:f(u,\sigma)=s} d(x_l , \beta(u, \sigma)) \right)}$$

And let s' denote the minimizing value of the previous state and u' the minimizing value of the channel symbol u. Thus the best path into the current state s passes through the previous state s' and is forced by channel symbol u'.

Given the path map $u^{l-1}(s')$ to the best previous state s', from the extended path map

$u^l(s) =(u^{l-1}(s'),u')$ giving the best (minimum distortion) path through the trellis from the initial state into the current state s at level l. In other words, for each current state find the distortion resulting from extending the K best paths into the previous states into the current state. The best is picked and the distortion and cumulative path indices to that state saved.

3. If l< L-1, set l+1$l + 1 \to l$ and go to 1. If l = L-1, pick the final state s_j yielding the minimum of the K values $D_L(s)$. The optimal path map is then $u^L(s_f)$.

Table 7.3: Viterbi Algorithm Trellis Encoder

A key facet above is that we have a block code with block length L, but we do not have to search a block code of 2^L by computing all 2^L possible distortions. Instead we compute a sequence of K distortions, where K is the number of states, and retain the running distortion for each state along with the accumulated distortions into each state and the accumulated channel symbol sequence. In particular, the complexity grows with the number of states, not with L.

The algorithm is implemented by keeping track of the following at each level of the trellis: (1) the best path into each node, and (2) the cumulative distortions up to that node. The principal of optimality means that knowing the best path into a node is equivalent to knowing at each state the best possible predecessors state. When the final node is reached, the path map the one which produces the smallest path distortion at the final depth.

Trellis coded quantization is a form of trellis coding that labels the trellis branches with subsets of production symbols. The approach was motivated by trellis-coded modulation. The novel feature of TCVQ is the partitioning of an expanded set of vector quantization symbols into subsets and the labeling of the trellis branches with these subsets. Labeling the trellis branches with properly formed subsets (instead of individual reproduction symbols) leads to a considerable reduction in encoding complexity.

7.2 Design of Trellis Coded VQ

7.2.1 Encoder/Decoder Structure

There is a simple way in which a VQ of dimension *k* and rate *R* is different from a TCVQ of the same dimension and rate. The VQ codebook contains 2^{kR} code vectors, and the VQ may represent any source any source vector any of the available code vectors. The TCVQ, on the other hand, has 2^{kR+m} code vectors, m a positive integer (i.e. for m=1, twice that of the VQ), but only a subset of size 2^{kR} of these code vectors may be used to represent a source vector at any instant of time. The TCVQ's structure is almost identical to that of the TCQ, except the latter operates on individual, as opposed to blocks of, source samples in its basic operations. The decoder is simpler to describe than the encoder, and that's what we treat first.

7.2.2 TCVQ Decoder

The TCVQ decoder is a finite-state machine with a state space S = { si : i = 0,1,…, K-1}, a set of input symbols Y = {0,1}kR (i.e., the set of all binary words of length kR), a codebook C= {ci € Rk : i = 0,1,…, 2^{kR+1}-1}, a next-state function g : S x Y → S, and an output function d : S x Y → C. Specifically, the decoder's behavior is governed by the following equations:

$$S_{n+1} = g(S_n, Y_n) \tag{7.4}$$

$$\hat{x}_n = d(S_n, Y_n) \tag{7.5}$$

where n = 0,1,2,… is the time index, Sn is the decoder state at time n(S_0 is the initial state), Yn is the binary word generated by the encoder at time n, and $\hat{x}_n$ is the reproduction for the source vector X_n generated at time n. (We assume one source vector is generated in each unit of time. This amounts to one source sample per 1/k unit of time.) Alternatively, the TCVQ decoder may be described in terms of a state diagram as shown in Figure 7.5, or more usefully, a trellis diagram as shown in Fig. 7.4.

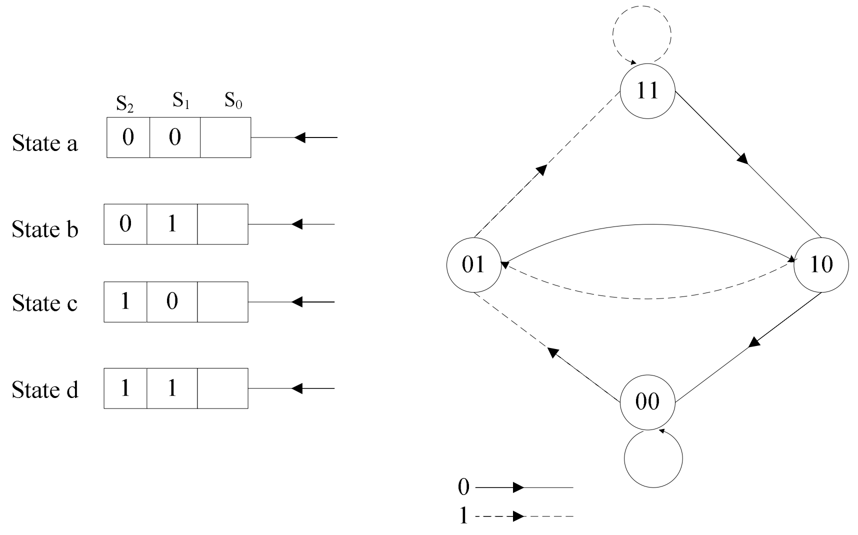


Figure 7.5: State diagram of TCVQ Decoder

The next-state and output functions, and hence the trellis, in the TCVQ decoder are of a special form. Suppose the decoder is in a given state at time n. Then it may receive any of the 2^{kR} possible binary words of length kR from the encoder. The first R' bits of the received word determine the decoder output at time n+1 and cause the decoder to move into that state. The remaining *kR - R′* bits determine the decoder output at time n. This means that a transition from one particular state to another one is caused by any of $2^{kR-R'}$ binary words of length kR. Such a transition is represented in the trellis diagram with a single branch, instead of $2^{kR-R'}$ parallel transitions. Hence there are $2^{R'}$ branches leaving each state and exactly that may enter it. With this convention, each branch may be labeled with a binary word of length R′ and a sub-codebook (a subset of *C*). Each sub-codebook is the set of code-vectors put out by the decoder in response to the last *kR - R′* bits of the received binary word from the encoder. Since each sub-codebook is of size $2^{kR-R'}$, there must be exactly $2^{R'+1}$ sub-codebooks that partition the overall codebook *C*.

7.2.3 TCVQ Enocder

The function of the TCVQ encoder is to find the best possible reproduction for any given sequence of source vectors. Specifically, for a search depth of L, the encoder looks for the sequence of code-vectors on the L-state trellis that is at minimum Euclidean distance from the given sequence of L source vectors. Each sequence of code-vectors on the trellis is specified by a path (a connected sequence of L branches) and the choices of code-vectors from the sub-codebooks associated with the branches. This requires LR′ bits for the first part and L(*kR - R′)* bits for the second part, which adds up to a total of LkR bits. For a given sequence of source vectors, the encoder uses the well-known Viterbi algorithm to find the best sequence of code-vectors. Then it releases all the LkR bits specifying this sequence at the end of the L stage trellis, which the decoder uses to generate the reproduction. The process is then repeated for the next L source vectors.

The encoding is accomplished in two steps.

Step 1) For each input vector s, find the closest code-word and corresponding distortion d_i, in each subset S_i.

Step 2) Let the branch metric for a branch labeled with subset Si be the distortion found in Step 1), and use the Viterbi algorithm to find the minimum distortion path through the trellis.

Complexity

The Viterbi algorithm (mse) encoding complexity of each TCVQ structure can be easily evaluated. There are $2^{R'k+m} / 2^{Rk}$ subsets, each of 2^{Rk-m} code-words. The first step in the encoding is to find the closest codeword in each subset to the source vector. For unstructured codebooks, such a full search requires

$k2^{kR-m}$ additions, $k2^{kR-m}$ multiplies, and $k2^{kR-m-1}$ two way comparisons, per subset. Next each trellis state has 2^m entering branches, with branch metrics the respective subset squared error. There are 2^m additions and 2^m-1 two-way compares to determine the "survivor" path at each trellis state. The total complexity is then

$k2^{(R+R')k}$ multiplies,

$k2^{(R+R')k} + N2^m$ additions,

$2^{R'k+m}(2^{Rk-m} - 1) + N(2^m-1)$ two way comparisons per k dimensional source symbol.

The TCVQ performance generally improves (for a fixed R) with an increase in k or N. Based on experimental evaluation of TCVQ encoding performance of a given source, it is possible to model the variations in mse as a function of k or N.

Thus, for a given level of complexity, “optimum” values of k or N can be selected for a given application.

The full search encoding complexity of k´-dimensional VQ is roughly

$k'2^{R\,k'}$ multiplies,
$k'2^{R\,k'}$ additions
$2^{Rk'}$ -1 two way comparisons
Hence, for the same dimension of encoding symbols, TCVQ is more complex than VQ.

7.2.4 Design Algorithm

The main goal of the partitioning step used in TCM is to provide subsets of channel symbols with maximal minimum distance within each subset. The overall set of channel symbols is partitioned in several stages such that a binary tree of subsets of channel symbols is generated. The two descendent nodes (or the subset at that nodes) of any internal node of the tree form a partition of the subset at that node. The subsets at the leaves are assigned to the trellis branches according to the “mapping by set partitioning” method. These observations suggest the following partitioning process for TCVQ:
Given an initial VQ codebook *C* of size M= 2^{kR+1}, the distances between all possible pairs of code-vectors are calculated and listed in a non-decreasing order along with the corresponding pairs. This gives a table, where the *i*th entry corresponds to code-vectors; c_i and that $c_{i'}$ are at distance d_i=c_i – c_i‘. First c_0 and $c_{0'}$ are placed in subsets A_0 and A_1, respectively, and the first entry is removed from the table. Then the following basic step is repeated as many times as necessary until the size of one of the subsets, A_0 and A_1, reaches M/2, at which point the remaining unassigned code-vectors (if any) are added to the other subset. The basic step involves looking for the entry with the smallest index *i* with one (but not both) of the two code-vectors already assigned to either A_0 or A_1,

followed by adding the unassigned code vector to the other subset. All entries with both code-vectors assigned are removed from the table. The basic step is described formally below:

1) Search the table to find an index j such that for all i < j, neither c_i nor c'_i belong to $A_0 \cup A_1$, but at least one of c_j and c'_j belongs to $A_0 \cup A_1$.

2) If both c_j and c'_j belong to $A_0 \cup A_1$, i.e. they are both already assigned then let $c_i \leftarrow c_{i+1}$, $c'_l \leftarrow c'_{i+1}$, for all i ≥ j. Go to 1).

3) If c_j belongs to A_0 (or A_1), then add c'_j to A_1 (or A_0). If c'_j belongs to A_0 (or A_1), then add c_j to A_1 (or A_0). If c'_j to A_1 (or A_0).

4) If the size of A_0 (or A_1) reaches M/2, then add the remaining unassigned codevectors (if any) to A_1 (or A_0) and stop. Otherwixe, go to 1).

To partition the overall VQ codebook *C* of size *M* into $2^{R'+1}$subsets, the algorithm is first applied to C to partition it into two subsets of size 2^{kR}, followed by applying it to these tow subsets to generate four subsets of size 2^{kR-1}, and so on.

7.3 TCVQ Results

7.3.1 Block delayed decision encoder and decoder

It begins with an initial state, finds an effective sequence of L=3 channel symbols for L input vectors, and then releases the L channel symbols to the channel. It then proceeds to do the same things for the next group of L input vectors. Typically the scheme will drive the encoder and decoder state back into the initial state s* at the end of one group of L input vectors. Here state ‘0’ is taken as initial state. By resetting the state prior to encoding each block, there is no memory of prior block used in encoding the current block. Hence, in this case the code behaves like a structurally constrained vector qunatizer of length Lk where k is the input vector dimension. This algorithm is illustrated in Figure: 6.10. The trellis diagram is considered for 4 state trellis. The current state and next state function for this trellis is shown in table 7.4. In this figure for block length of three, the path map 000 has lowest distortion for given sequence of 3 vectors.

The decoder is set at 000 after each block. In the same way we can do the encoding and decoding using 8 state trellis.

Current state		*Next state*	
0	0	0	*1*
1	1	2	*3*
2	2	0	*1*
3	*3*	*2*	*3*

Table 7.4 Next state function 4 state trellis

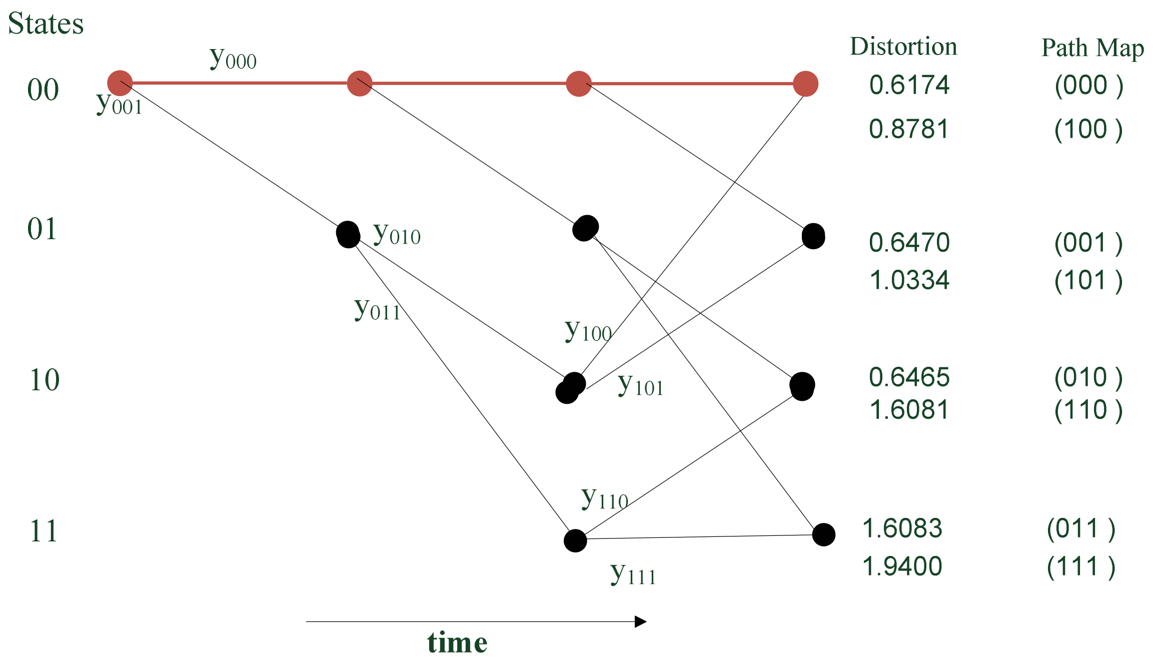


Figure: 7.6 4 state trellis diagram for blocked delayed decision encoder

TCVQ encoder and decoder for 4-state and 8-state trellis is designed of different VQ dimensions for a uniform and a zero-mean, unit-variance, i.i.d. Gaussian source using a training sequence of 100000 samples. This required designing several VQ codebooks. The codebook is designed using maja_ma_ne.wav file and the kem_chho.wav file is taken as testing sequence.

For 4 state and 8 state trellis, the simulation results is shown in table 7.5 and 7.6 respectively. The same thing can be visualized from Figure 7.7 and Figure 7.8 respectively. The codebook size is kept constant so with increasing the VQ dimension, the SQNR is reduced. SQNR with Gaussian source is 1 dB higher than with uniform source.

Source		*Vector dimension*					
		1	2	3	4	5	*6*
Uniform	MSE	0.004027	0.0048452	0.0086526	0.0072436	0.0069007	*0.011105*
	SQNR	8.413	7.6096	5.0913	5.8632	6.0738	*4.0074*
Gaussian	MSE	0.0038655	0.004018	0.006455	0.0049868	0.0051532	*0.0070305*
	SQNR	*8.5907*	*8.4227*	*6.3638*	*7.4846*	*7.342*	*5.9929*

Table 7.5 SQNR vs. VQ for 4 state trellis-block delayed

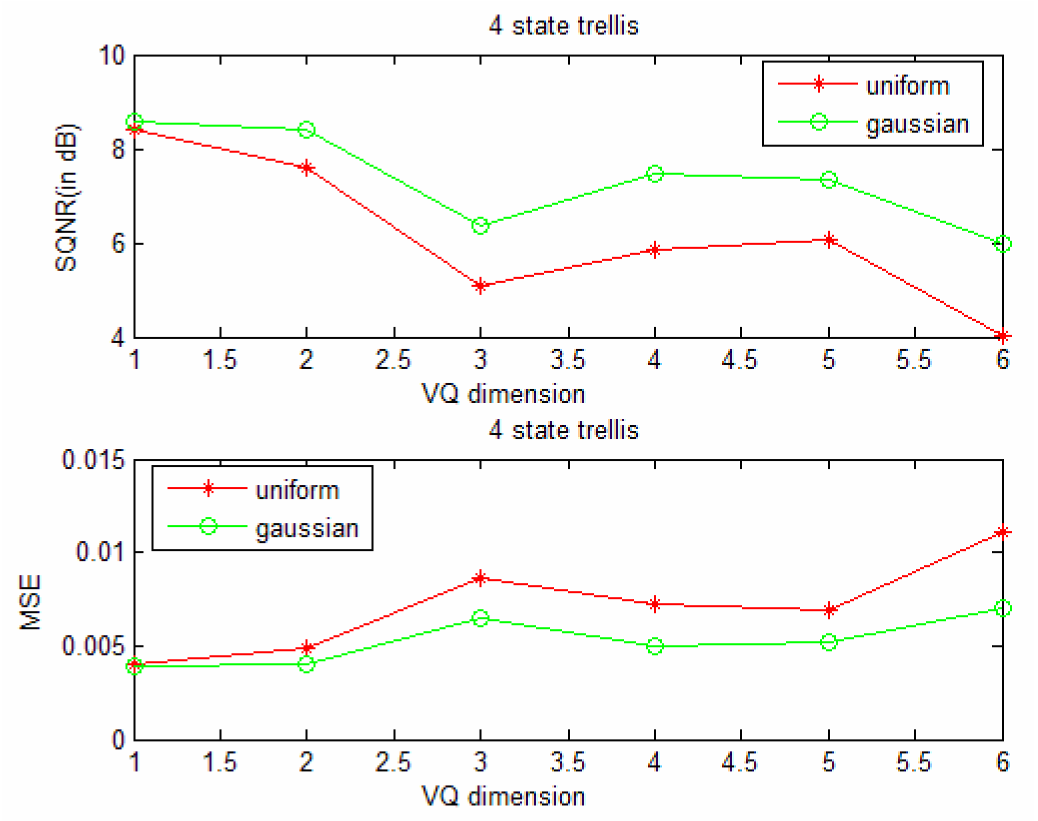


Figure 7.7 SQNR vs. VQ for 4 state trellis- block delayed

Source		*Vector dimension*					
		1	2	3	4	5	*6*
Uniform	MSE	0.0040589	0.0046617	0.0089728	0.0073027	0.0069073	*0.011071*
	SQNR	8.3787	7.7773	4.9335	5.828	6.0697	*4.0211*
Gaussian	MSE	0.0037788	0.003928	0.0064263	0.0049829	0.0051474	*0.0070273*
	SQNR	*8.6892*	*8.5211*	*6.3832*	*7.488*	*7.3469*	*5.9949*

Table 7.6 SQNR vs. VQ for 8 state trellis -block delayed

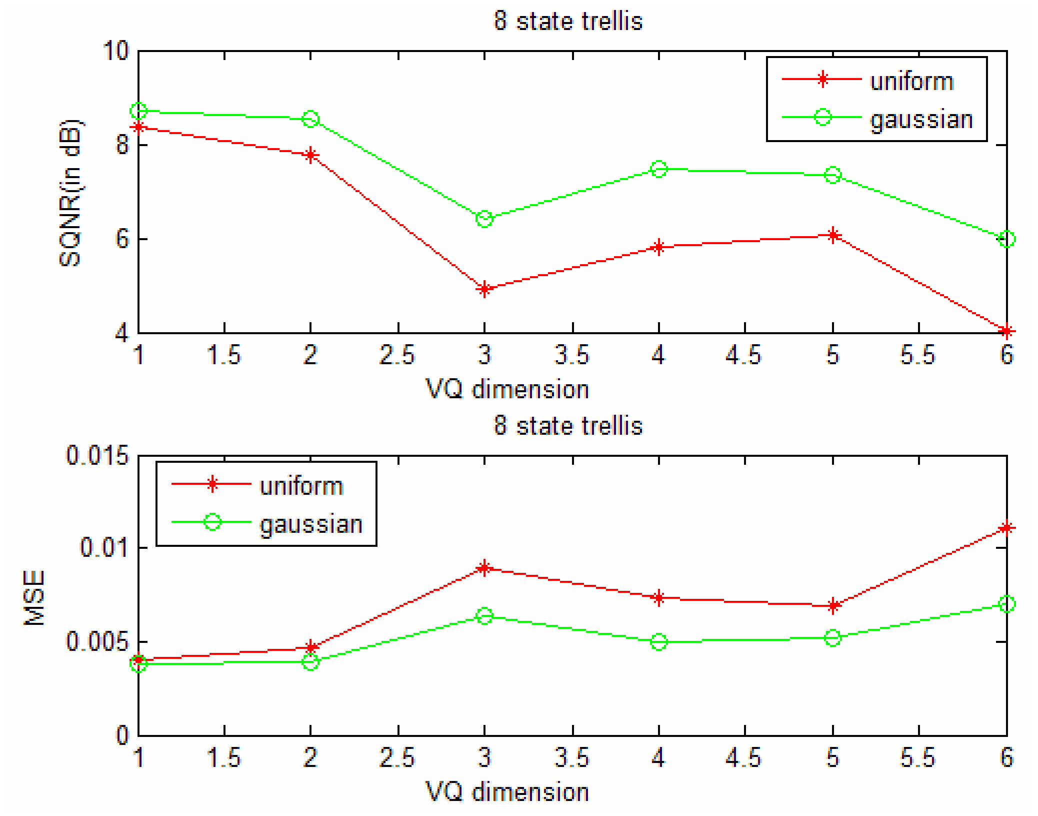


Figure 7.8 SQNR vs. VQ for 8 state trellis-block delayed

Now, the codebook is generated from full 512×512 ‘lena’ image which contains 8-codevectors of dimension 1×2.The same image is coded and decoded with block delayed decision coder. Figure 7.9 shows results for the same. PSNR of 18.201 dB is obtained. PSNR vs. VQ dimension plot for 4 state and 8 state trellis is shown in Figure 7.10.

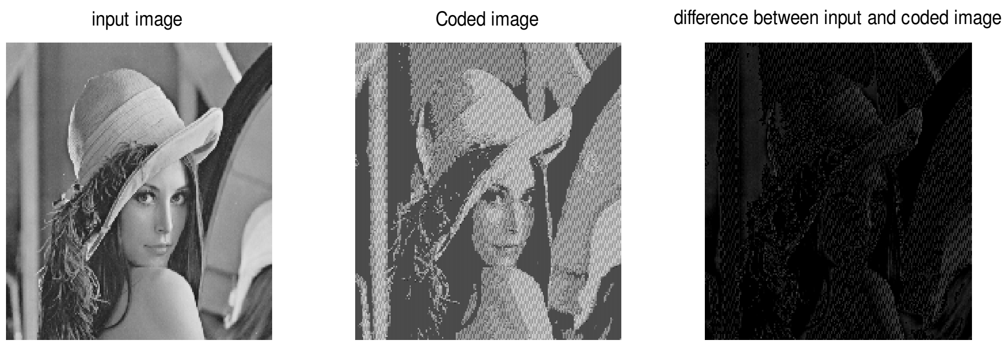


Figure 7.9 1x2 TCVQ with block delayed decision coder

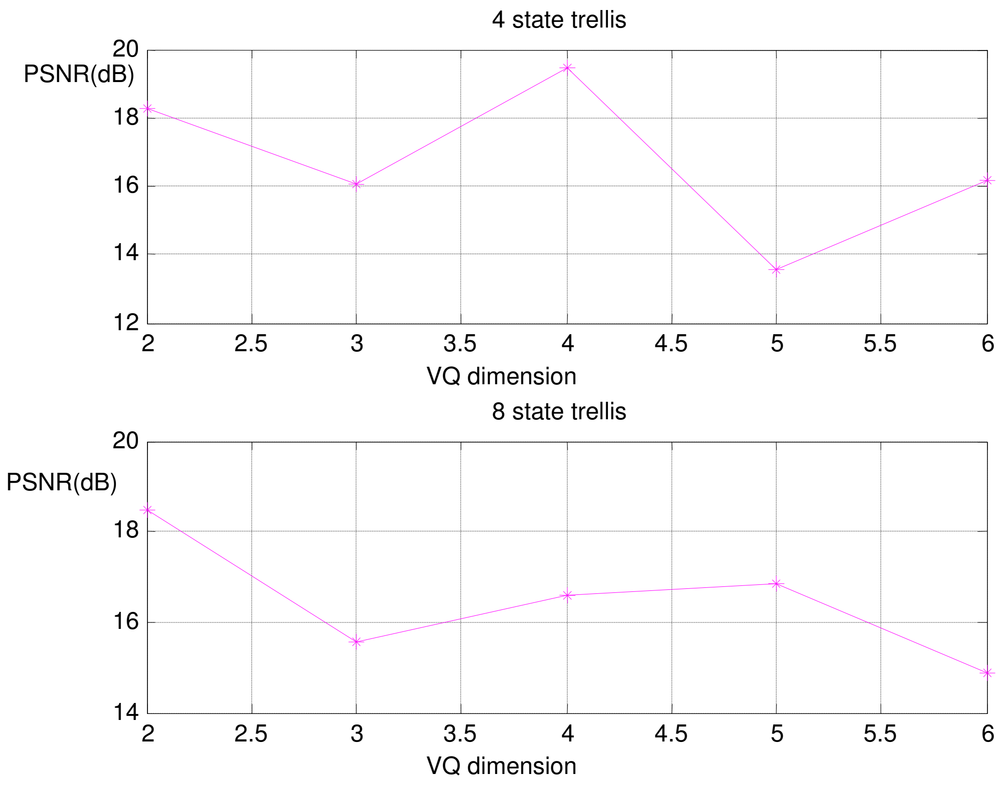


Figure 7.10 PSNR vs. VQ dimension

7.3.2 Continuous encoding & Decoding

Here the path map is selected as in the block case, but instead of driving the encoder and decoder to initial state s* at the end of one block of L input vectors, the encoder and decoder begins at the last state of previous block. After encoding of fist block, the entire trellis comes in picture. The encoder then again searches ahead L vectors so as to pick a new path and advanced further. Such incremental encoding provides smoother operations and fewer blocking effects in the data.

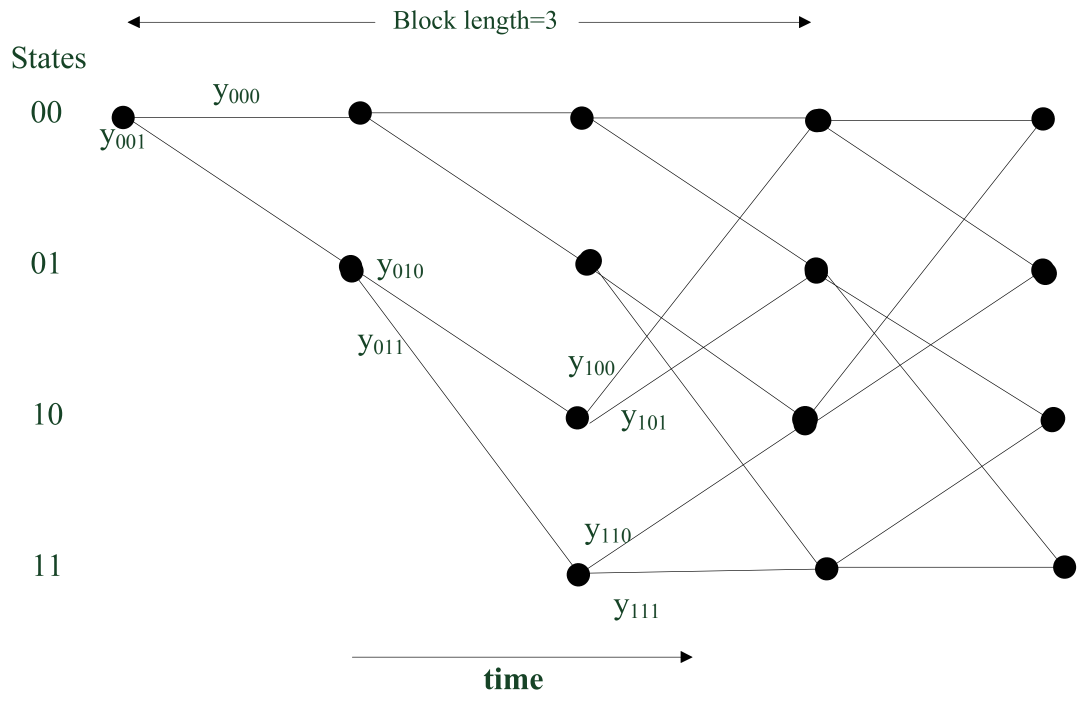


Fig.: 7.11 4 state trellis diagram for continuous encoder

Source		*Vector dimension*				
		2	3	4	5	*6*
Uniform	MSE	0.0072648	0.0092347	0.0068773	0.0066746	*0.011586*
	SQNR	5.8505	4.8085	6.0886	6.2185	*3.8236*
Gaussian	MSE	0.0052828	0.0064575	0.0049947	0.0051678	*0.007017*
	SQNR	*7.2341*	*6.3621*	*7.4777*	*7.3297*	*6.0012*

Table 7.7 SQNR vs. VQ for 4 state trellis –continuous

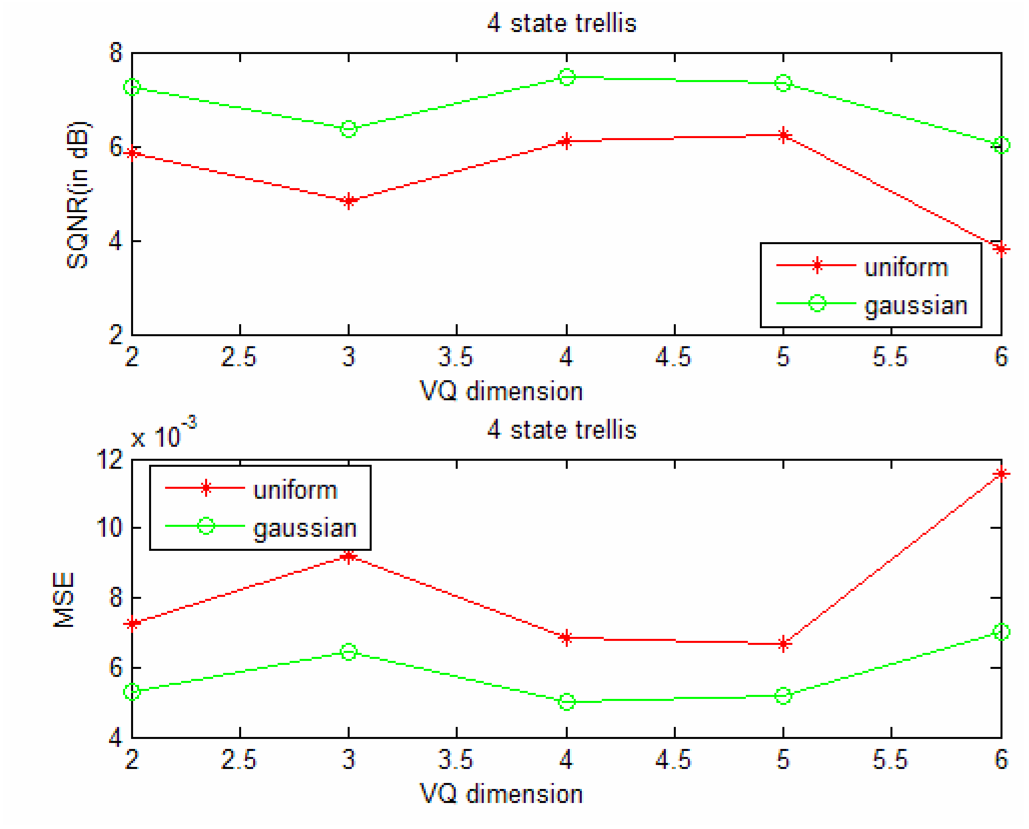


Figure 7.12 SQNR vs. VQ for 4 state trellis–continuous

Source		*Vector dimension*				
		2	3	4	5	*6*
Uniform	MSE	0.0063529	0.0098531	0.0073874	0.0066498	*0.011601*
	SQNR	6.433	4.527	5.7779	6.2347	*3.8179*
Gaussian	MSE	0.0058754	0.0072088	0.0050154	0.0051696	*0.0070327*
	SQNR	*6.7724*	*5.8842*	*7.4597*	*7.3282*	*5.9916*

Table 7.8 SQNR vs. VQ for 8 state trellis-continuous

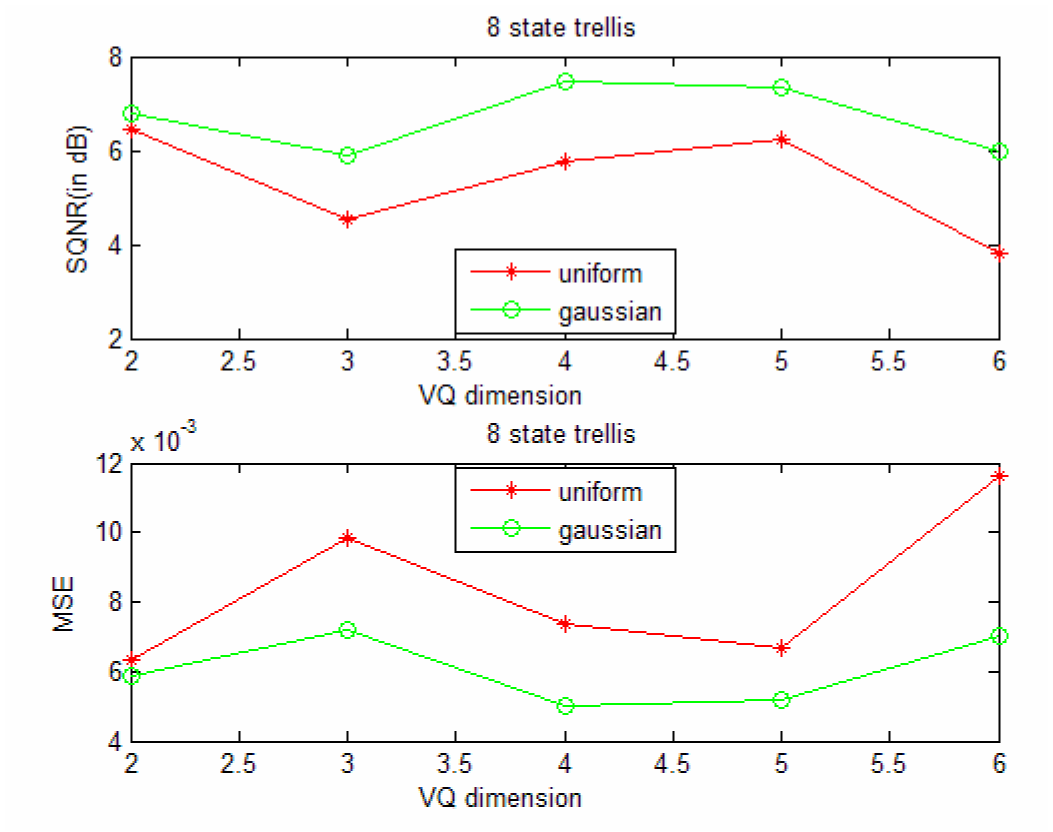


Figure 7.13 SQNR vs. VQ for 8 state trellis-continuous

Now, the codebook is generated from full 512×512 ‘lena’ image which contains 8-codevectors of dimension 1×2.The same image is coded and decoded with continuous coder. Figure 7.14 shows results for the same. PSNR of 18.9166 dB is obtained.

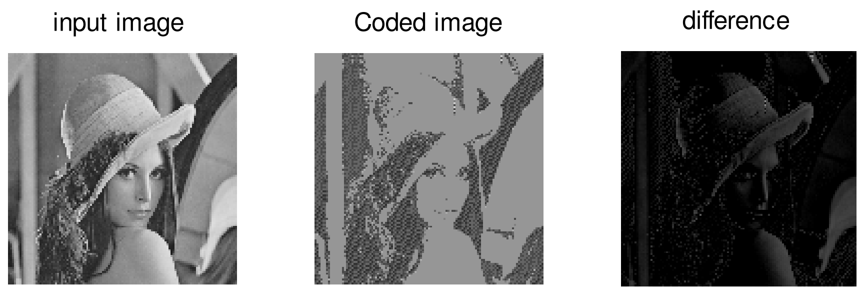


Figure 7.14 1x2 TCVQ with continuous coder

7.3.3 Viterbi Algorithm

The trellis diagram for N=4 state is shown in the Figure 7.15 Given the survivor paths ending at time n can be determined as follows. Let $d_{n-1}^{i}(x_n y_n)$ be the overall distortion related to the survivor path ending at node i at time n-1. The updated survivor path ending at node k at time n is determined by first finding, for each branch entering state k at time n, the best subset codeword that minimizes the distortion between the input vector x_n and the reproduction vector y_n. Then we compute the overall distortion associated with each of the two possible paths to node k at time n and select the path with the minimum distortion as the updated survivor path ending at node k. As shown in Figure 7.15 the path with brown color shows the survivor path (minimum distance path) at each node. After all N survivor paths at time n are determined, the time index is incremented and the process is repeated until a certain depth. In Figure 7.15 the path with solid brown color shows the survivor path for block length of 3.

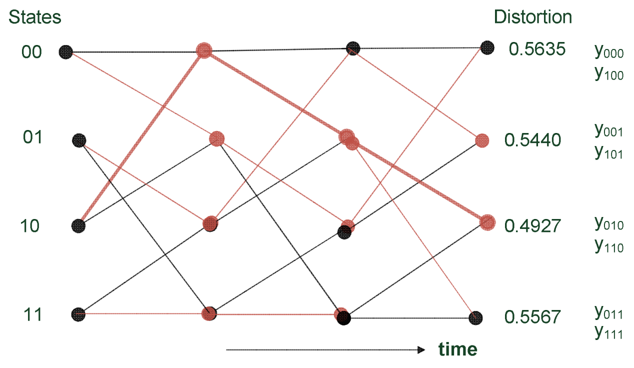


Figure 7.15 Viterbi encoding

Results for 4 and 8 state trellis are in Table 7.9, Table 7.10. They are also depicted in Figure 7.16, Figure 7.17.

Source		*Vector dimension*					
		1	2	3	4	5	*6*
Uniform	MSE	0.0038039	0.0091421	0.010083	0.012979	0.012925	*0.013614*
	SQNR	8.6605	4.8523	4.4269	3.3305	3.3484	*3.1229*
Gaussian	MSE	0.0036939	0.0071339	0.0075187	0.0076989	0.0076163	*0.0075481*
	SQNR	*8.7879*	*5.9295*	*5.7013*	*5.5985*	*5.6453*	*5.6844*

Table 7.9 SQNR vs. VQ for 4 state trellis-viterbi

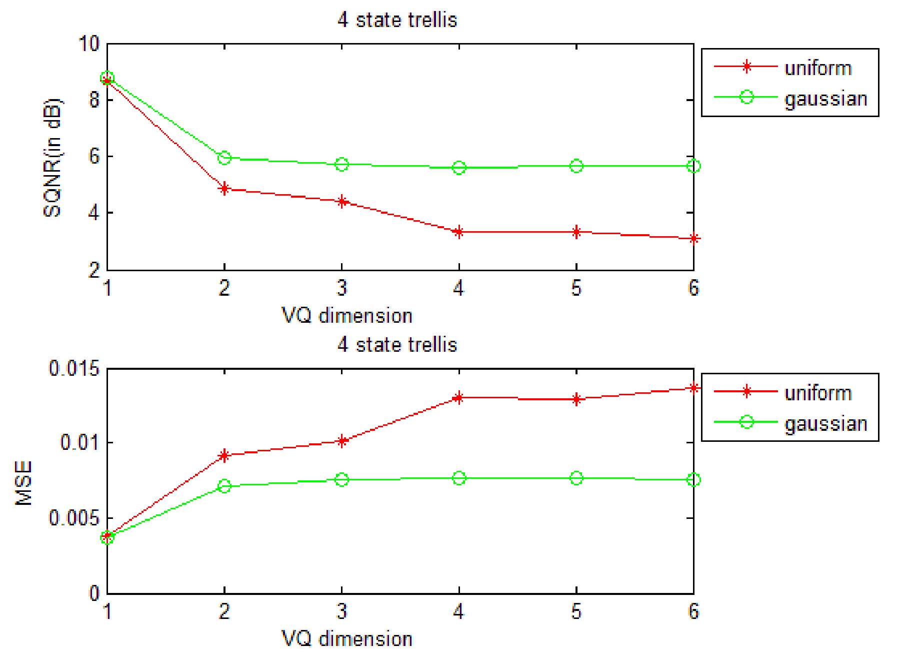


Figure 7.16 SQNR vs. VQ for 4 state trellis-viterbi

Source		*Vector dimension*					
		1	2	3	4	5	*6*
Uniform	MSE	0.0038001	0.0091387	0.017999	0.012968	0.012916	*0.013665*
	SQNR	8.6648	4.8539	1.9104	3.334	3.3514	*3.1068*
Gaussian	MSE	0.0036946	0.0071305	0.0074446	0.0076887	0.0076077	*0.007543*
	SQNR	*8.7871*	*5.9316*	*5.7443*	*5.6043*	*5.6502*	*5.6873*

Table 7.10 SQNR vs. VQ for 8 state trellis-viterbi

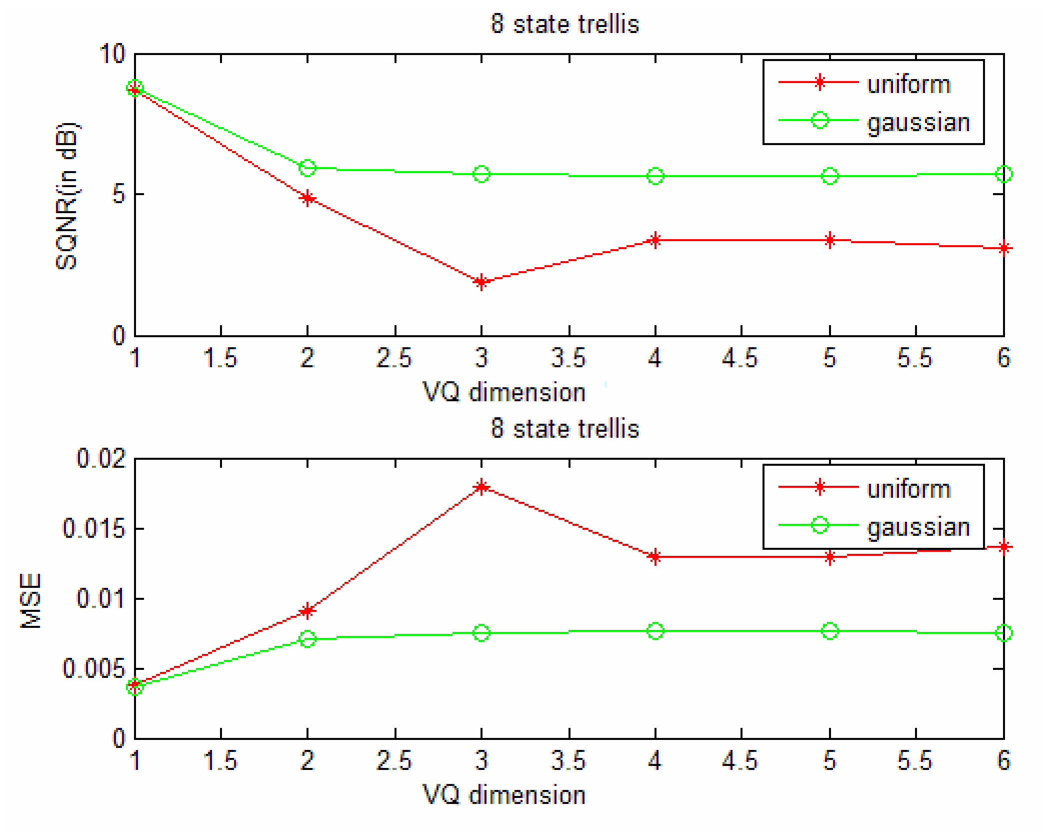


Figure 7.17 SQNR vs. VQ for 8 state trellis-viterbi

7.4 Summary

This chapter has outlined basics of trellis coded vector quantization (TCVQ) and how to produce vector reproduction symbols. The design of TCVQ coders is considered, with partition and branch labeling rules presented for memory-less vector sources. The complexity of viterbi algorithm for TCVQ is outlined. Design of TCVQ for 4 state and 8 state trellis is discussed with three different methods. Among the three methods of TCVQ, the performance of block delayed decision encoder is better than other methods. If the source is Gaussian, SQNR is higher than with uniform source.

Chapter 8

Comparison of Various Techniques

8.1 Requirements of FSVQ

FSVQ requires full search on the codebook. Memory requirements are higher for better performance. This also increases the processing time. Processing time is highest compared to the other methods. The required codebook storage space in words and the search complexity (number of operations per input vector in an exhaustive codebook search) are both proportional to k N , where k is vector dimension and N is number of code words.

$$k\,N = k\,2^{rk} \tag{8.1}$$

Resolution r is measured in bits per vector component. Memory and complexity grows exponentially with dimension.

8.2 Requirements of TSVQ

In tree-structured VQ (TSVQ), the search is performed in stages. If the codebook size is $N=m^d$, then d m-ary search stages are needed to locate the chosen code vector, d represents the depth of the tree.

The number of search operations at each stage is proportional to m since each test is an exhaustive nearest neighbor search through a set of m test vectors. Thus, the total search complexity is proportional to md rather than m^d where the proportionality constant depends on the vector dimension k.

On the other hand, the storage requirement of TSVQ is increased compared to FSVQ or other unstructured VQ. In addition to storing m^d code vectors (the leaves of the tree); the test vectors for each node of the tree must also be stored. There is one node at the first stage, m nodes at second stage, m^2 nodes at the third stage, etc. Hence, the total number of nonterminal nodes is $1 + m + m^2 + \ldots + m^{d-1} = (m^d - 1) / (m - 1)$. Since each non terminal node stores m code or test vectors , the total number of k-dimensional vectors to be stored including code vectors is $m(m^d - 1) / (m - 1)$.

8.3 Requirements of MSVQ

In MSVQ, the encoding task is divided into successive stages. Note that the complexity is reduced from $N = \prod_{i=1}^{m} N_i$ to $\sum_{i=1}^{m} N_i$ and the equivalent product codebook is generated from the Cartesian product $K_1 \times K_2 \times \ldots \times K_m$. Thus both the complexity and storage requirements is greatly reduced .

As usual, there is a performance penalty with this product code technique. Overall quantization error between input and output is equal to the quantization error introduced in the last stage. The signal to quantizing noise power ratio in dB (SNR) for the multistage quantizer is given by

$$SNR = \sum_{i=1}^{m} SNR_i \tag{8.2}$$

8.4 Requirements of TCVQ

The trellis consists of the set of all paths starting at the root node and traversing one of the K state nodes at each time instant and terminating at a fixed node at time L-1. L is the encoder delay or search depth. We do not have to search a block code of 2^L by computing all 2^L possible distortions. Instead we compute a sequence of K distortions. The complexity grows with the number of states, not with L.

8.5 Comparison of Various Techniques

The Comparison of various techniques is shown in figure 8.1.

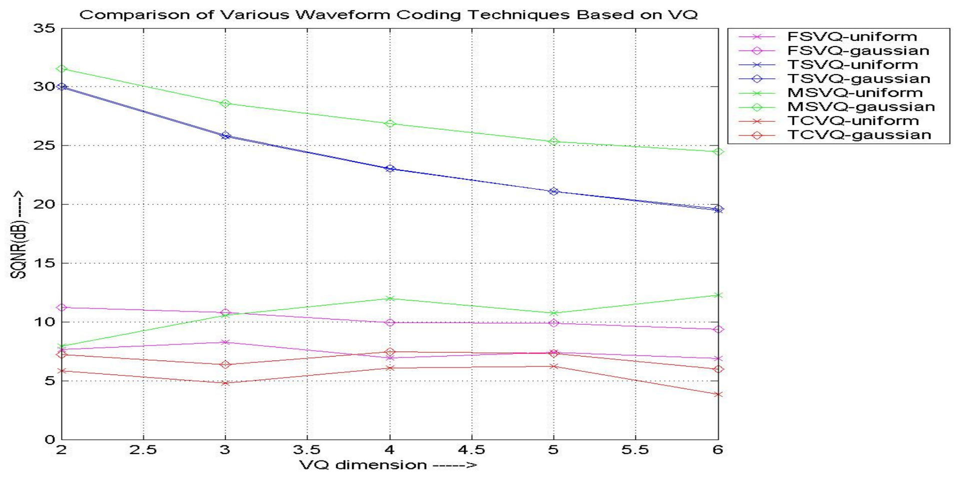


Figure 8.1: Performance of Various Techniques

Comparing the performance, MSVQ, TSVQ gives much better performance compared to FSVQ, TCVQ. MSVQ is better by 5.3835 and 15.1443 dB compared to Full Search VQ with Uniform and Gaussian pdf respectively. TSVQ is better by 5.7796 and 10.2383 dB compared to Full Search VQ with Uniform and Gaussian pdf respectively. Full Search VQ is better by 3.0729 and 3.3635 dB compared to TCVQ with Uniform and Gaussian pdf respectively. The benchmark for these observations is Full Search VQ .All other techniques are compared with Full Search VQ.

8.6 Summary

There is no better performance of various VQ techniques compared to FSVQ. Searching makes prohibitive use of FSVQ. If the storage requirements are not stringent then TSVQ can be used. The complexity and storage requirements reduce in MSVQ but at the cost of SQNR. Computational requirements are minimum in TCVQ, but encoding delay is more.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

The major conclusions drawn from the available results are as follows:

(i) Code books with Gaussian pdf compared to uniform pdf give better performance in case where FSVQ is applied to speech. SQNR 3.5707 dB more is be obtained in case of speech.

(ii) As the number of code words is increased SQNR and hence PSNR also gets better in case where FSVQ is applied to an image.

(iii) In case where FSVQ is applied to image, code books with Gaussian pdf compared to uniform pdf does not produce better results. This indicates the random distribution of pixel amplitudes in the image.

(iv) As the tree depth increases the SQNR gets better in case of TSVQ applied to speech.

(v) Code books with Gaussian pdf compared to uniform pdf gives generally better performance in case where TSVQ is applied to speech. SQNR improvement of maximum 0.102 dB is obtained.

(vi) Code books with Gaussian pdf compared to uniform pdf gives better performance in case where TSVQ is applied to image. PSNR improvement of maximum 6.981 dB is obtained.

(vii) As the tree depth increases the SQNR gets worse in case of TSVQ applied to image. This indicates the random distribution of pixel amplitudes in the image.

(viii) With MSVQ applied to speech, maximum SQNR improvement of 4.732 , 12.867 and 23.6025 dB is obtained for the three different speech signals respectively.

(ix) Code books with Gaussian pdf compared to uniform pdf give better performance in case where TCVQ is applied to speech.

(x) Performance improvement in SQNR from 4-state trellis to 8-state trellis is maximum 0.0984 dB (block-delayed), 0.0058 dB (viterbi) for Gaussian pdf.

9.2 Future Work

(i) Apart from TSVQ , MSVQ , TCVQ other VQ techniques can be used.

(ii) If the pdf of a signal is known in advance by estimation, then codebook can be prepared accordingly.

(iii) The waveform coding techniques are applied by assuming noiseless channel. The techniques can be applied in noisy channels.

(iv) The techniques can be applied to video.

(v) Hardware realizations using VLSI / DSP can be done.

Chapter 10
Bibliography

[1] J. MacQueen. ”Some methods for classification and analysis of multivariate observations,” In *Proc. Of the Fifth Berkely Symposium on Math.Stat. and Prob.* , vol.1 ,pp. 281 -296,1967.

[2] R.G.Gallager, *Information Theory and Reliable Communication*.John Wiley & Sons , New York ,1968.

[3] T. Berger, *Rate Distortion Theory*. Prentice-Hall Inc.,Englewood Cliffs , New Jersey ,1971.

[4] R. M.Gray, *Source Coding Theory*. Kluwer Academic Press , Boston,1990.

[5] J. T. Tou and R.C. Gonzales, *Pattern Recognition Principles*. Addison-Wesley, Reading, Mass., 1974.

[6] W.H. Equitz, “Fast algorithms for vector quantization picture coding,” in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, pp. 725-728, 1987.

[7] W.H. Equitz, “A new vector quantization clustering algorithm,” *IEEE Trans. Acoust. Speech Signal Process*, pp. 1568-1575, October 1989.

[8] J. Ward, “Hierarchical grouping to optimize an objective function,” *Journal of American Statistical Assoc.*, pp. 37:236-244, March 1963.

[9] M.R. Anderberg, *Cluster Analysis for Applications*. Academic Press, San Diego, 1973.

[10] Y. Linde, A. Buzo, and R.M. Gray, “An algorithm for vector qunatizer design” *IEEE Trans. Comm.*, vol. COM-28, pp.84-95, January 1980.

[11] E. Diday and J.C. Simon, Clustering analysis. in K.S. Fu and K.S. Fu, editors, *Digital pattern recognition*, Springer – Verlag, NY 1976.

[12] S.P.Lloyd, ”Least squares quantization in PCM,” Unpublished Bell Laboratories technical note. Portions presented at the Institute of Mahtematical Statistics meeting Atlantic City New Jersey September 1957. Published in the March 1982. Special issue on quantiztion of the *IEEE Transaction on Information Theory*, March 1957.

[13] D.T.S.Chen, “On two or more dimensional optimum quantizers,”in Proc. *International Conference on Acoustic, Speech and Signal processing*, pp. 640-643, Hartford, CT, 1977.

[14] E.E. Hilbert, “Cluster compression algorithm: a joint clustering/data compression concept,” Jet Propulsion Lab Publication, Pasadena, Calif., pp. 77-43, December 1977.

[15] J. Vaisey and A Gersho, “Simulated annealing and codebook design,”in *Proc. of the International Conference on Acoustics, Speech. And Signal Processing*, pp.1176-1179, New York, April1988.

[16] K. Zeger and A. Gersho, “A stochastic relaxation algorithm for improved vector quantizer design.,”Electronics Letters, vol. 25 ,pp. 896-898, July 1989.

[17] A.E. Cetin and V. Weeracody, ”Design vector quantizers using simulated annealing,” in *Proc. of the International Conference on Acoustic, Speech and Signal Processing* , p. 1550 , December 1988.

[18] K. Rose, E. Gurewitz, and G.C. Fox, ”A deterministic annealing approach to clustering,” Pattern Recognition Letters, vol.11 , pp.589-594, 1990.

[19] K. Rose, E. Gurewitz, and G.C. Fox, ”Statistical mechanics and phase transitions in clustering,” Physical Review Letters, vol. 65, pp. 945-948, 1990.

[20] J.C. Dunn, “A fuzzy relative of the ISODATA process and its use in detecting well-separated clusters,” Journals of cybernetics, vol. 3, pp.32-57, 1974.

[21] J.C. Bezdek, ”A convergence theorem for the fuzzy ISODATA clustering algorithm,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 3, pp. 1 – 8, 1980.

[22] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, *Classification and regression trees*. Wadsworth, Belmont, California 1984.

[23] J. Makhoul, S. Roucos, and H. Gish, ”Vector quantization in speech coding,”in *Proc. IEEE*, vol. 11, pp. 1551-1587, Nov. 1985.

[24] X. Wu and K. Zhang., ”A better tree-structured vector quantizer,” in J. A. Storer and J. H. Reif, editors, proc. data compression conference, IEEE Computer Society Press, pp. 392-401, Snowbird, Utah, April 1991.

[25] E. A. Riskin, ”Optimal bit allocation via the generalized BFOS algorithm,” *IEEE Trans. Inform. Theory*, vol. 37, pp. 400-402, March 1991.

[26] B. Ramamurthi and A. Gesho,”Classified vector quantization of images,” *IEEE Trans. Comm.*, vol. COM-34, pp. 1105-1115, November 1986.

[27] K. Aizawa, H . Harashimia, and H. Miyakawa ,"Adaptive vector quantization of picture signals in discrete cosine transform domain," *Electronics and Communication in Japan*, Part I, p.70, 1987.
[28] R. A. King and N. M. Nasrabadi, "Image coding vector quantization in the transform domain," *Pattern recognition letters*, vol. 1, pp.323-329, 1983.

[29] S. Adlersberg and V. Cuperman, "Transform domain vector quntization for speech signals," in *Proc. International Conference on Acoustics, Speech and Signal Processing*, Vol. 4, pp. 45.4.1-45.4.4, Dallas, Texas, April 1987.

[30] H. Abut and S.A. Luse," Vector quantizers for subband coded waveforms," in *Proc. International Conference on Acoustics, Speech and signal processing,* vol. 1, pp.10.6.1.-10.6.4, San Diego, California, March 1984.

[31] A. Gersho, T. Ramstad, and I.Versvik, "Fully vector –quantized subband coding with adaptive codebook allocation," in *Proc. International Conference on Acoustics, Speech, and signal Processing*, vol. I, pp. 10.7.1-4, March 1984.

[32] P.H.Westrink, D.E. Boekee, J. Biemond, And J. W. Wooda, "Subband coding of images using vector quantization," *IEEE Trans. Comm.*, vol. COM-36, pp. 713-719, June 1988.

[33] M. Antonini, M. Barland, P. Mathieu and I.Daubechies, "Image coding using vector quantization in the wavelet transform domain," in *Proc. of the International Conference on Acoustic, Speech and Signal processing*, pp. 2297-2300, Albuiquerque, April 1990.

[34] M. J. Sabin and R. M. Gray, "Product code vector quantizers for wave-form and voice coding," *IEEE Trans. Acoustics, Speech and Signal Processing.*, vol. ASSP-32 , pp. 474-488, June 1984.

[35] A. Buzo, A.H. Gray Jr., R.M. Gray and J.D. Markel, "Speech coding based upon vector quantization," *IEEE Trans. Acoustics, Speech and Signal Processing, vol.* ASSP –28, pp. 562 – 574, October 1980.

[36] C. Tsao and R.M. Gray, "Shape- gain matrix quantities for LPC speech," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-34 , pp.1427-1439, December 1986.

[37] C.F. Barnes and R.L. Frost, "Necessary conditions for the optimality of residual vector Quantizers," in abstract of the 1990 *IEEE International Symposium on Information Theory*, age 34, San Diego, Calif., January-1990.

[38] R.L. Frost,C.F. Barnes, and F. Xu, "Design and performance of residual quantizers," in J.A. Storer and J.H.Reif,editors, *Proc. Data Compression Converence* , Snowbird,Utah, IEEE Computer Society Press, pp.129-138, April 1991.

[39] W.Y. chan and A. Gersho, " Constrained – storage vector qunatization in high fidelity audio transform coding," in *Proc. of the International Conference on Acoustic, Speech and Signal processing*, Toronto, Canada, May 1991.

[40] A. Gersho and Y. Shoham, "Hierarchical vector quantization of speech with dynamic codebook allocation," in *Proc. of the International Conference on Acoustics,Speech, and Signal Processing*, pp.10.9.1-10.9.4, San Diego,March 1984.

[41] Y.S. Ho and A Gersho, "Variable – rate contour – based interpolative vector quantization for image coding," in conference record: *IEEE Global Communication Conference*, pp.1890-1893, Nov. 1988.

[42] Y. S. Ho and A. Gersho, "Classified transform coding of image using interpolative quantization," in proc. of the *International Conference on Acoustics, Speech, and Signal Processing,* pp.1890-1893, May 1989.

[43] Y.S. Ho and A Gresho, "A variable rate image coding scheme with vector quantization and clustering interpolation," in proc. 1989 *Globecom*, pp. 25.5.1-35.5.1, 1989.

[44] Y.S. Ho and A Gersho, "A pyramidal image coder using contour based interpolative vector quantization," in proc. *SPIE International Symposium On Visual and Image processing* , vol. 1199, pp. 733-740, Nov. 1989.

[45] J.H. Conway and N.J. A. Salone, "Voronoi regions of lattices second moments of polytopes and quantization," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 211-226, March 1982.

[46] J.H. Conway and N.J. A. Salone, *Sphere packing, lattices and groups*. Springs – Verlag, New York, 1988.

[47] J.P. Adoul and M. Barth, "Nearest neighbor algorithm for spherical codes from the Leech lattice," *IEEE Trans. Inform. Theory,* vol. IT-34, pp. 1188-1202, 1988.

[48] T.R. Fischer,"A pyramid vector quantizer," *IEEE Trans. Inform. Theory*, vol. IT – 32, pp. 568-583, July 1986.

[49]G.D.Forney,Jr.,"Multidimensional constellations Part II:Voronoi Constallations," *Journal of Selected Areas in Communications*,vol. 7, pp. 941-958,1958.

[50] G.D. Forney Jr., ”Coset codes- Part II: Introduction and geometrical classification,” *IEEE Trans. Inform.Theory*, vol. IT-34, pp.1152-1187,1989.

[51] J.L. Bentley,”Multidimensional binary search trees used for associative searching,” *Comm. ACM*, pp.109-226, September 1975.
[52] J.H.Friedman, F.Baskett, and L.J. Shustek, ”An algorithm for finding nearest neighbors,” *IEEE Trans. on Computers*, vol. C-24(10), pp.1000-1006,October 1975.

[53] A. Lowry, S. Hossian and W. Millar,”Binary search trees for vector quantization,” in *proc. of the International Conference on Acoustics, Speech and Signal processing*, pp. 2206-2208, Dallas 1987.

[54] B. Ramasubramanian and K. K. Paliwal, “An optimized k-d tree algorithm for fast vector quantization of speech,” in *Proc. European Signal Processing Conference*, pp.875-878. Grenoble, 1988.

[55] V. Cuperman and A Gersho,”Vector predictive coding of speech at 16kb/s,” *IEEE Trans. Comm.*, vol. COM-33, pp.685-696, July 1985.

[56] P.C. Chang and R.M. Gray,”Gradient algorithm for designing predictive vector quantizers,” *IEEE Trans. Acoustics, Speech and Signal process.*, vol. ASSP – 34, pp.769-690, August 1986.

[57] J. C. Kieffer, ”Stochastic stability for feedback quantization schemes,” *IEEE Trans. Inform. Theory*, vol. IT-28, pp.248-254, March 1982.

[58] T. Fine,”Properties of an optimal digital system and applications,” *IEEE Trans. Inform. Theory, vol.* IT-10, pp.287-296, Oct 1964.

[59] B. McMillan,”Communication systems which minimize coding noise,” *Bell System Tech. Journal*, vol. 48(9), pp. 3091-3112, Nov 1969.

[60] G.Gabor and Z.Gyorfi, *Recursive Source Coding.* Springer-Verlag,New York,1986.

[61] R. M. Gray, a. Buzo,Y. Matsuyama , A.H. Gray Jr., and J.D. Markel, ”Source coding and speech compression,” in *proc. of the International Telemeteing Convergence*, vol. XIV, pp. 871-878. Los Angeles, Calif., Nov 1978.

[62] A. Buzo, A.H. Gray Jr., R.M. Gray and J.D. Markel, “Optimal quantizations of coefficients vectors in LPC speech,” in *Proc. International Conference on Acoustics, Speech, and Signal processing*, pp.52-55, Washington, D.C. April 1979.

[63] A. Buzo, A.H. Gray Jr., R.M. Gray and J.D. Markel, ”Speech coding based upon vector quantization,” *IEEE Trans, Acoustics, Speech and Signal processing* , vol. ASSP –28, pp. 562 – 574, October 1980.

[64] D. Wong, B. H. Juang and A. H. Gray Jr., “An 800 bit/s vector quantization LPC vocoder,” *IEEE Trans, Acoustics, Speech and Signal processing*, vol. ASSP-30, pp.770-779, October 1982.

[65] G. Rebolledo, R. M. Gray, and J. P. Burg, ”A multirate voice digitizer based upon vector quantization*,” IEEE Trans. Comm.*, vol. COM-30, pp. 721-727, April 1982.

[66] J.P. Adoul and P. Mabillean,”4800 bps RELP vocoder using vector quantization for both filter and residual representation,” in *Proc. International Conference on Acoustics, Speech, and Signal Processing,* vol.1, p.601, Paris, April 1982.

[67] N. S. Jayant and P. Noll, *Digital Coding of waveforms*. Prentice Hall, Englewood Cliffs, New Jersey, 1984.

[68] R.G. Gallanger, *Information Theory and Reliable communication*. John Wiley & Sons, New York, 1968.

[69] J.T. Tou and R.C. Gonzales, *Pattern Recognition Principles*. Addision-Wesley, Readind, Mass., 1974.

[70] P.H. Westernick, J. Biemond, and D.E. Boekee, ”An Optimal bit Allocation Algorithm for sub-band coding,” in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, pp. 757-760,1988.

[71] F. Kossentini, M. J. T. Smith, and C. F. Barnes, “Image coding using entropy-constrained residual vector quantization,"*IEEE Trans. Image Processing*, vol. 4, no. 10, pp. 1349-1357, Oct. 1995.

[72] W. P. LeBlanc, B. Bhattacharya, S. A. Mahmoud, and V.Cuperman, “Efficient search and design procedures for robust multi-stage VQ of LPC parameters for 4 kb/s speech coding," *IEEE Trans. Speech and Audio Proc.*, vol.1, no. 4, pp. 373-385, Oct. 1993.

[73] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Boston, MA: Kluwer, 1992.

[74] B. H. Juang and A. H. Gray Jr., " Multiple stage vector quantization for speech coding," in *Proc. International Conference on Acoustics, Speech, and Signal Processing*, vol.1, pp.597-600, Paris, April 1982.

[75] R.L. Frost, C.F. Barnes, and F. Xu., "Design and performance of residual quantizers," in J.A. Storer and J.H.Reif,editors, *Proceeding Data Compression Converence,* IEEE Computer Society Press, pp.129-138,Snowbird,Utah,April 1991.

[76] Makwana M. V., "Design and Implementation of Tree Structured Vector Quantization", *M.E. Dissertation*, Gujarat University, June 2007.

[77] Brahmbhatt P. J.,"Design and Implementation of Trellis Coded Vector Quantizer", *M.E. Dissertation*, Gujarat University, August 2009.