EXPLOITING TEMPORAL AND SPATIAL REDUNDANCIES FOR VECTOR QUANTIZATION OF SPEECH AND IMAGES

A Thesis
Presented to
The Academic Faculty

by

Chu Meh Chu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2015

EXPLOITING TEMPORAL AND SPATIAL REDUNDANCIES FOR VECTOR QUANTIZATION OF SPEECH AND IMAGES

Approved by:

Professor David V. Anderson,
Committee Chair
School of ECE
Georgia Institute of Technology

Professor David V. Anderson, Advisor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Professor Justin K. Romberg
School of ECE
Georgia Institute of Technology

Professor Elliot Moore II
School of ECE
Georgia Institute of Technology

Professor Thomas Morley
School of Mathematics
Georgia Institute of Technology

Professor Anthony Yezzi
School of ECE
Georgia Institute of Technology

Date Approved: November 11,2015

To my parents and to God who gives me the grace to do all things

ACKNOWLEDGEMENTS

An African proverb goes “It takes a village to raise a child”. My journey to this particular juncture in my life has been colored by many people, and punctuated by many phases of growth in my academic, emotional, and spiritual growth that makes the aforementioned statement not only valid but real. I remain forever grateful to all those who have been in each of those phases of my life. A special thanks goes to my advisor, Dr. David V. Anderson, who has always believed in me and encouraged me to be disciplined and open-minded when it comes to approaching research and life in general. I particularly thank him for being a man of fiath who always leaves every body whom he meets more encouraged and hopeful. I also thank my peers at the Efficient Signal Processing Laboratory- Nathan Parrish, Bradley Whiteley, Kaitlin Fair, Brandon Carroll, Dr. Devangi Parikh, and Dr. Kenneth Chiu- with whom I have had the delight of discussing challenging academic rigors as well as provided support to each other when needed.

I would be remiss if I don’t thank the various groups and entities who have sponsored my academic journey up till this point. First, I am grateful to the National Science Foundation Graduate Research Fellowship program for funding my first three years of graduate school. I am also thankful to the ECE department at Georgia Tech for allowing me to be a teaching assistant for two years. The McNair Achievement Program at my undergraduate school (NJIT) and the Summer Undergraduate Research Experience at Georgia Tech provided me a safe platform to develop my research confidence and I am forever grateful to them.

To my parents-Dr. Thomas Chu and Veronica Meh Chu- whom this dissertation is dedicated to. I could not have done this endeavor without their unwavering support, love and vote of confidence. I thank them from instilling in me from early on a genuine love

of learning balanced with play so that I could someday rest on the shoulders of academic giants. To my siblings- Dr. Geh, Buh, Ndjuoh, Enseng, and Zuh- I thank you all for always checking on me and for the side discussions and the animated vacations you all made possible during my tenure as doctoral student. Their serenity during this period has been exemplary. I have also had an adopted family in Atlanta by the Tandongfor's. Both Ambrose and William Tandongfor have been instrumental in making me feel at home in Atlanta. I thank these two men for their kindness and selflessness.

I have been blessed with a plethora of friends from all walks of life. I cannot name them all but I will be remiss if I don't thank Divine Mekande Ngome, Bonam Sakwe, Chiluwata Lungu, Dr. Uzoma Onunkwo, Dr. Edem Wornyo, Dr. Brett Matthews, Innocent Wamey, and Didier Melone. Each of these people have encouraged and pushed me to be a better person in ways I could not have done by myself.

I have been blessed immensely by two church communities in Atlanta: the Georgia Tech Catholic Center and the Cameroon Community of Catholics. These two communities have been my spiritual anchors and have given me the much needed confidence and fellowship to become a better human being in service of others. A special word goes out to Fr. Mario Di Lella, Fr. Timothy Hepburn, Fr. Kevin Peek, and Fr. Henry Atem, and the musical choirs whom I have sang and played instruments for. I am grateful for them all.

I must recognize the support of ASA, OMED, the Christian Order of Dynamic Engineers (CODE), Sacred Heart Ex-Students Association Network (SHESANS) for all the relationships that were cultivated and nurtured through service and felloship within all these organizations. A heartfelt thanks to Ms. Lillie McPhee at OMED for always having a listening ear. I also will like to thank all the other professors I have done a research project for including Dr. Nikil Jayant, Dr. Yucel Altunbasak, and Dr. May Wang. Each of these professors provided a learning platform from which my intellectual growth could blossom.

Last but not least, I thank God for His every flowing mercy and grace that allowed me to persist through all the hurdles to witness this special moment. His grace alone is sufficient

indeed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of the proposed research is to *compress data such as speech, audio, and images using a new re-ordering vector quantization approach that exploits the transition probability between consecutive code vectors in a signal*. Vector quantization is the process of encoding blocks of samples from a data sequence by replacing every input vector from a dictionary of reproduction vectors. Shannon's rate-distortion theory states that signals encoded as blocks of samples have a better rate-distortion performance relative to when encoded on a sample-to-sample basis. As such, vector quantization achieves a lower coding rate for a given distortion relative to scalar quantization for any given signal. Vector quantization does not take advantage of the inter-vector correlation between successive input vectors in data sequences. It has been demonstrated that real signals have significant inter-vector correlation. This correlation has led to vector quantization approaches that encode input vectors based on previously encoded vectors. Some methods have been proposed in literature to exploit the dependence between successive code vectors. Predictive vector quantization, dynamic codebook re-ordering, and finite-state vector quantization are examples of vector quantization schemes that use inter-vector correlation. Predictive vector quantization and finite-state vector quantization predict the reproduction vector for a given input vector by using past input vectors. Dynamic codebook re-ordering vector quantization has the same reproduction vectors as standard vector quantization. The dynamic codebook re-ordering algorithm is based on the concept of re-ordering indices whereby existing reproduction vectors are assigned new channel indices according a structure that orders the reproduction vectors in an order of increasing dissimilarity. Hence, an input vector encoded in the standard vector quantization method is transmitted through a channel with new indices such that 0 is assigned to the closest reproduction vector to the past

reproduction vector. Larger index values are assigned to reproduction vectors that have larger distances from the previous reproduction vector. Dynamic codebook re-ordering assumes that the reproduction vectors of two successive vectors of real signals are typically close to each other according to a distance metric. Sometimes, two successively encoded vectors may have relatively larger distances from each other. Our likelihood codebook re-ordering vector quantization algorithm exploits the structure within a signal by exploiting the non-uniformity in the reproduction vector transition probability in a data sequence. Input vectors that have higher probability of transition from prior reproduction vectors are assigned indices of smaller values. The code vectors that are more likely to follow a given vector are assigned indices closer to 0 while the less likely are given assigned indices of higher value. This re-ordering provides the reproduction dictionary a structure suitable for entropy coding such as Huffman and arithmetic coding. Since such transitions are common in real signals, it is expected that our proposed algorithm when combined with entropy coding algorithms such binary arithmetic and Huffman coding, will result in lower bit rates for the same distortion as a standard vector quantization algorithm. The re-ordering vector quantization approach on quantized indices can be useful in speech, images, audio transmission. By applying our re-ordering approach to these data types, we expect to achieve lower coding rates for a given distortion or perceptual quality. This reduced coding rate makes our proposed algorithm useful for transmission and storage of larger image, speech streams for their respective communication channels. The use of truncation on the likelihood codebook re-ordering scheme results in much lower compression rates without significantly distorting the perceptual quality of the signals. Today, texts and other multimedia signals may be benefit from this additional layer of likelihood re-ordering compression.

CHAPTER I

INTRODUCTION

Television, and telephone signals before the age of digitization were all transmitted as analog signals. Digitization of analog signals such as speech, image and radar have proliferated the communication domain with the advent of modern processors that can handle digital data much more easily. This digitization is advantageous as it allows for more manipulation, communication, and compression. Scalar quantization (SQ) is the process of assigning each sample to a finite amplitude. Scalar quantization is used widely in speech communication standards such as ITU G.711, ITU G.722, ITU G.726, ITU G.727, but these standards achieve good sound quality by using relatively high coding rate of 48kbps. Vector quantization is based on the rate-distortion principle that as the dimension of a vector increases towards infinity, the coding rate of a given data sequence approaches a minimal coding rate bound for a given distortion. Hence, the coding rate per sample reduces as the dimension increases.

Other forms of vector quantization capture the correlation between vectors but do not exploit the inherent correlation that exists in the encoded vectors of a given data sequence. Dynamic codebook re-ordering vector quantization (DCRVQ) is a vector quantization approach to explore redundancy in the vector-quantized data sequence. The DCR approach attempts to model correlation as a function of the distance between dictionary vectors of the vector quantizer codebook. This aforementioned approach falls short because the distance measure is a static metric and does not change if we have a fixed vector dictionary codebook. Hence, once a dictionary codebook is obtained, no prior information about the transition between two vectors is used to further lower the coding rate.

Our algorithm, *likelihood codebook re-ordering vector quantization (LCR)*, exploits

this transitioning between the dictionary vectors in a data sequence. It is observed that the transition between two dictionary vectors exhibit non-uniform probability distribution function if the transition is treated as a random variable. The contributions of this dissertation are as follows:

1. Introduction of the likelihood codebook re-ordering method to losslessly transmit data following the vector quantization stage

2. Introduction of a truncation approach to LCR to achieve more significant coding gains

3. Application of lossless coding method such as Huffman coding on the re-ordered indices

4. Application of the LCR method to both speech and image data

5. Introduction of the an online and adaptive likelihood codebook re-ordering scheme that yields more reduction in coding gains

The rest of the dissertation is arranged as follows: Chapter 2 highlights in detail the other vector quantization schemes and their shortcomings that propel us to propose our new algorithm. Chapter 3 elaborates on the contributions of our new method. Chapter 6 and Chapter 7 discuss the application of our method to speech and image datasets respectively. Chapter 9 concludes the results and findings of the dissertation.

CHAPTER II

BACKGROUND OF VECTOR QUANTIZATION

Data compression is a field that has been of primary importance because of the increased need for transmission of digital data [25]. In the past, data was transmitted in analog format as in telephone services, analog television, and the use of cameras prior to digital cameras [18]. Data is obtained in digital format in the two consecutive processes of sampling and quantization. Sampling involves acquiring the signal in question at specified times while quantization is the process of assigning pre-defined values from a given set to these signal samples. The quantization stage just described is often called scalar quantization and is an essential building block for analog-to-digital conversion. Essentially, an N point scalar (one-dimensional) quantizer, $Q : \mathbb{R} \rightarrow C$ where $\mathbb{R}$ is the real line and $C = \{y_1, y_2, y_3, \ldots, y_N\} \in \mathbb{R}$ is the codebook of size N.

The output values, y_i, are sometimes referred to as reproduction values. In differential encoding such as differential pulse code modulation, the difference signal between the input signal between the input signal and its predicted version is encoded instead of the actual input signal [33]. This difference signal has a reduced range compared to that of the actual signal. This reduction in effect yields a smaller bit rate for the same signal-to-noise ratio. Differential encoding exploits redundancy between successive scalar values of a data sequence. This is greatly employed in speech processing applications, and is included in coding standards. Adaptive pulse code modulation [2, 36, 41] adjusts the size of the quantization step to allow further reductions of the coding rate for a given signal-to-noise ratio. Speech pulse code modulation and its variants have been used in many coding standards such as G.711, G.726, G.722, subband coding, voice over internet protocol (VoIP), DVD and blu-ray players [17].

Vector quantization is an extension of scalar quantization with the difference that it encodes groups of samples at a time. Vector quantization is based on Shannon rate distortion theorem. As the name implies, a vector of samples is encoded one sample at a time and is represented by one of several dictionary vectors from a designed codebook. Shannon's rate-distortion theorem states that as the dimension of a vector increases and approaches infinity, the rate-distortion function approaches the Shannon theoretical bound. The Shannon theoretical bound represents the minimum rate that can achieved for a given distortion or the converse. The fruit of this principle is that vector quantization achieves lower compression even if the source is memory-less, that is, the source is a sequence of identically and independently distributed random variables.

In this chapter, we present the principles of vector quantization and the performance metrics and/or criteria of evaluation. An information-theoretic presentation is also made to highlight the importance of Shannon's rate-information theory.

2.1 Memory-less vector quantization

Memory-less vector quantization was proposed in [28] to achieve a better compression rate for a given distortion. This method is the foundational vector quantizer algorithm. In this method, a vector of samples is represented by a dictionary or codebook of reproduction vectors that divide the training vector space into separate Voronoi regions. The method proposed by Gray, *et al.*, assumes that the reproduction vector has the smallest distance to a given input vector relative to other reproduction vectors in a codebook.

In this approach, each input vector is encoded independently of the previous data vector. In this subsection, the concepts, properties, and design for memory-less vector quantization which are elaborated in [28] are discussed.

2.1.1 Quantization

Every compression algorithm has an encoder and a decoder. Let us assume that an input vector $\mathbf{x} \in \mathbb{R}^k$ and an encoder α assigns to each input vector $\mathbf{x} = \{x_0, x_1, \ldots, x_{k-1}\}$ a

channel symbol in some set **M** and a decoder β maps every channel symbol u in **M** in a reproduction alphabet $\hat{A}$. The channel symbol is often assumed to be a space of 2^R binary R-dimensional vectors. The reproduction alphabet may consist of real vectors of a different dimension.

If **M** has M elements, then the quantity $R = \log_2 M$ is the rate of quantizer in bits per vector and $r = R/k$ is the rate in bits per symbol or when input is sampled waveform, it represents bits per sample.

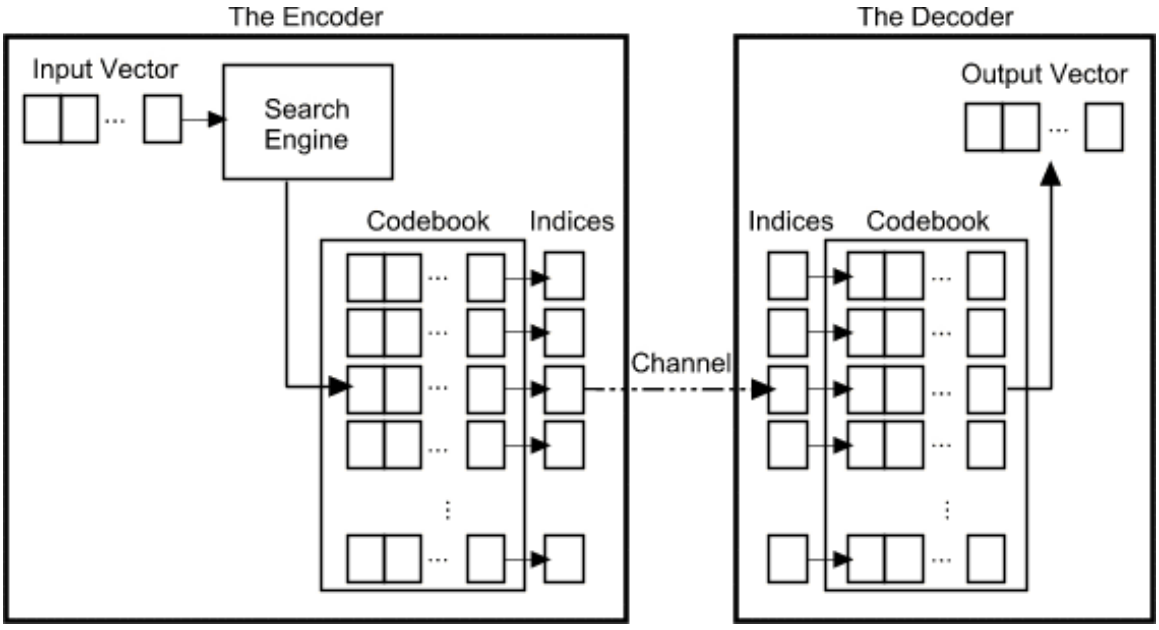


Figure 1: Vector quantization block diagram

The use of the quantizer system for data compression is illustrated in Figure 1. The input data vectors can be parameter vectors in voice coding system, consecutive vectors in image coding or features of biomedical images [19]. For the sake of simplicity, it is assumed that the channel is noiseless. While real samples are rarely noiseless, error correction coding for noisy channels can effectively provide a noiseless transmission of the VQ symbols. In short, the emphasis on the noiseless channel is to focus on the data compression and not reflect any real model. The goal of this memory-less scheme is to produce

the "best" possible reproduction code for a given rate R. The idea of "best" is incomplete unless understood with reference to the distortion model.

2.1.2 Distortion

The distortion measure, d, is a cost function $d(x, \hat{x})$ of reproducing any input vector $\hat{x}$ from x where k is the dimension of the vector. With such a metric, we can quantify the performance of a system by an average distortion between the input and final reproduction vectors. Thus, a system is considered good if it produces a small average distortion. There are two principal distortion measures to characterize fidelity of a reconstructed signal to a given input signal. The distortion metric is computed typically in one of two ways as described subsequently.

2.1.2.1 Squared error distortion measure

Here, the input and reproduction spaces are k-dimensional euclidean spaces. The distortion

$$d(x, \hat{x}) = \|x - \hat{x}\| = \sum_{i=0}^{k-1} (x_i - \hat{x}_i)^2$$

is the distortion between input vector, x, and reproduction vector, $\hat{x}$, where x_i, $\hat{x}_i$ are the i^{th} components of the actual signal vector and reproduction vector respectively. This measure is commonly used for waveform coding. There are many times when the weighted euclidean distance is used to quantify the distortion as shown below

$$d(x, \hat{x}) = \|x - \hat{x}\| = \sum_{i=0}^{k-1} w_i (x_i - \hat{x}_i)^2$$

where w_i is the individual weight per given vector dimension. It is also very common to measure performance by the squared-error-distortion

$$SNR = 20 \log_{10} \frac{E\|X\|}{E(d(X, \hat{X}))}$$

.

2.1.3 Computational complexity

For a fixed coding resolution rate per unit vector component, the performance of vector quantization increases as the dimension k increases [3]. This performance improvement with dimension can be explained by recognizing that the statistical independence between signal samples are increasingly exploited. The improvement in performance also comes from the sphere packing gain. The required codebook storage space in words and search complexity are both proportional to kN where k is the vector dimension and N is the codebook size. N is typically equal to 2^{rk} where r is the bits per vector sample. An illustration of how the complexity of a vector scheme changes with dimensionality is shown in table 2.1.3 below.

Table 1: Table of complexity and operation speed as function of dimension

Dimension	Complexity	Operations per vector
1	2	16k
4	64	128k
8	2048	2M
10	10240	8M
12	49152	33M

It is easy to discern from the above table that complexity increases as dimensionality increases. Sometimes constraints are placed in the codebook structure such that computational complexity is reduced. These constraints oftentimes imply an increase in distortion. The following subsection describes common approaches of exploiting structure in the codebook while reducing computational complexity.

2.1.4 Constrained Vector Quantization

The vector quantization schemes that fall under this category have a reduced computational cost at the price of a slightly increased distortion measure.

2.1.4.1 *Tree-structured Vector Quantization*

Tree structured vector quantizers were first proposed by Buzo, *et al.*, [7] and are a bi-product of decision trees based algorithm. In tree structured vector quantization (TSVQ), the codebook is designed using a tree structure whereby each layer down the tree node eliminates the need to search the non-traversed notes. The prime advantage is to reduce the computational search cost. For an m-ary codebook of size $N = m^d$ where d is the number of layers in the codebook. Tree-structured VQ used md units to search the TSVQ codebook unlike m^d for unstructured VQ. Overall, the distortion performance is poorer relative to unconstrained VQ for a given codebook size.

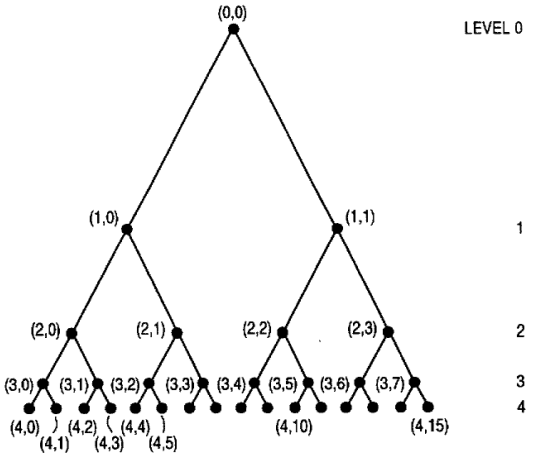


Figure 2: Tree-structured vector quantization

2.1.4.2 *Product-code Vector Quantization*

An approach to reduce computational complexity for encoding data vectors is product code vector quantization. In this approach, an input vector is decomposed into vectors of smaller dimensions such that each sub-vector is encoded separately with a different codebook. This method [37] yields much reduced computational complexity though it may be suboptimal in performance. This reduction in coding distortion is a result of the fact that the sub-vectors

typically have vector-to-vector dependencies which are not exploited when the input vector is decomposed into smaller sub-vectors.

2.1.4.3 Mean-removed Vector Quantization

In mean-removed VQ [5], the mean of an input vector is removed, followed by the quantization of the mean and the resultant vector called mean-removed vector separately. The technique is effective when source input vectors are similar to each other. For example, mean-removed VQ is used for a set of similar images with differing amounts of background illumination. The effect of the varying lighting conditions can be effectively reduced by removing the mean of each before quantization. This method is an example of product vector quantization where the vector is partitioned into a scalar and vector.

2.1.4.4 Gain-shape Vector Quantization

In applications such as speech, where the dynamic range of the source input is quite large, gain-shape VQ [37] may be used. For such sources, a very large codebook is needed to represent the various vectors from the source. This requirement is reduced through gain-shape VQ, in which the source input vectors are normalized by a suitable normalization factor. The normalized vector and the normalization factor are then quantized separately.

Memory-less VQ encodes each data vector individually and as such does not incorporate *a priori* knowledge of the previously encoded input vectors. In short, it does not use memory of previously quantized input vectors to encode the current vector. Subsequent research after the LBG algorithm demonstrated that performance can be improved by introducing memory into the vector quantization scheme [11]. In a nutshell, exploiting the correlation or redundancy that exists in previously encoded data vectors have been shown to reduce coding rate.

2.2 Vector Quantization with Memory

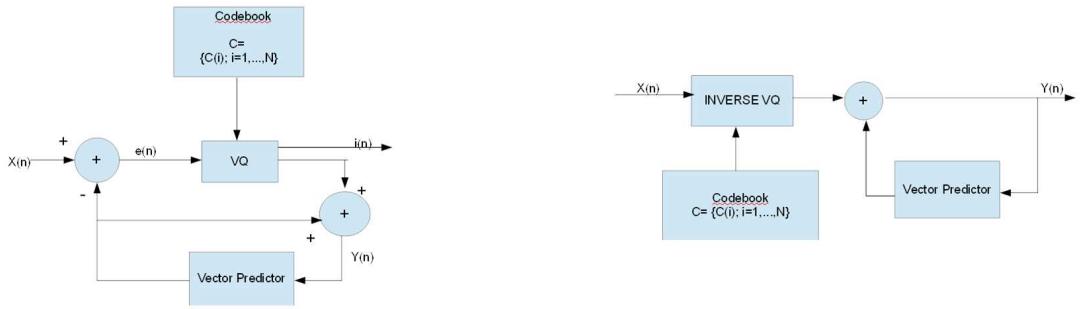
Vector quantization with memory is introduced in the this section. Memory-less vector quantization schemes such as LBG algorithms capture the correlation between dimensions within a single vector. Further compression can be achieved for a given distortion by exploiting the correlation between input data vectors. This can be done by incorporating memory into a vector quantization scheme by using different codebooks for each input vector. The codebooks change depending on the past data input vectors. Thus, vector quantization with memory is generally dubbed *feedback vector quantization*. Feedback vector quantization (FVQ) is an extension of scalar adaptive quantization with backward estimation [22]. In this section, 2 vector quantization schemes that encapsulate the idea of using memory are briefly described.

2.2.1 Vector Predictive Quantization

In [11], Gersho and Cuperman proposed a predictive vector coder (PVQ) that is a vector generalization of differential pulse-code modulation (DPCM) [34]. Differential pulse-code modulation is a scalar quantization algorithm in which the difference signal sample between the data sample and its predicted estimate from previous data samples are quantized using pulse-code modulation (PCM) [1]. The quantized difference signal has a much smaller distortion than that of quantizing the original signal. The result of this predictive vector quantization is a higher signal-to-noise ratio relative to LBG VQ. Figure 3 illustrates the working of the encoding and decoding using predictive vector quantization.

2.2.2 Finite-state Vector Quantization

Foster and Gray introduced in [15] another quantization scheme with memory called finite-state vector quantization (FSVQ). Unlike predictive vector quantization that has an infinite number of reproducible vectors, FSVQ has a finite set of reproducible vectors. Finite-state VQ uses a much simpler prediction rule than PVQ and hence is more computationally



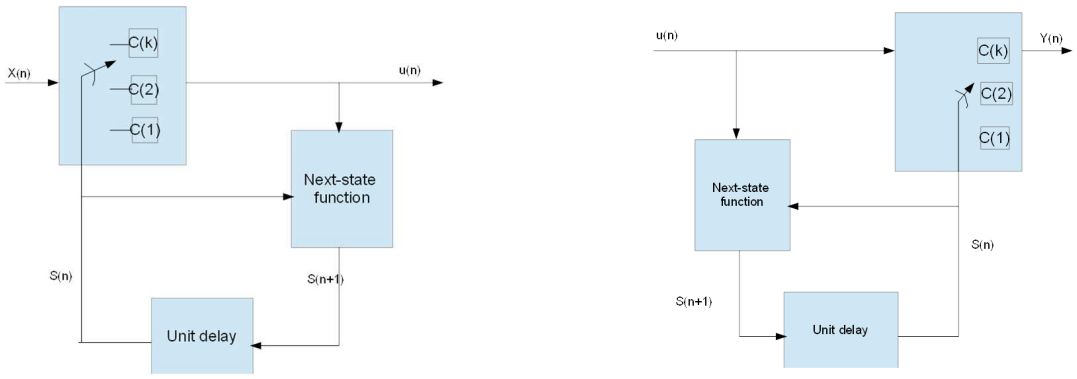
(a) (b)

Figure 3: (a) Predictive vector encoder and (b) decoder

efficient than PVQ for a given coding rate and distortion.

A finite-state vector quantizer can be viewed as a collection of K separate memory-less vector quantizers with a selection rule that determines which of the K codebooks is used to encode the current input vector into a channel index. The advantage of using FSVQ is that it explores the correlation between successive vectors by tracking its dependence on previous vectors by means of a state. The state is tracked in both the encoder and decoder and hence it is not encoded and quantized as the actual channel index is. Experiments on real data and Gauss-Markov signals have demonstrated that FSVQ outperforms VQ for a given coding rate in terms of signal-to-noise ratio (SNR) for sources with inter-vector correlation. FSVQ's encoder and decoder are shown in Figure 4.

We denote the state code books as $C_s = \{\beta(u, s); \text{all } u \in \mathbb{N}, s \in S\}$. The next-state function $f(u, s)$ is a function of the current state s and the index of the quantized input vector u. Both the state codebook and the next-state function are used to encode any input vector at a given time.



(a) (b)

Figure 4: Finite State Vector Quantization (a) Encoder and (b) decoder block diagram

2.3 Adaptive vector quantization

Memoryless vector quantization typically maps input vectors into one of many predefined code vectors. For a real signal, dictionary vectors are not identically and independently distributed. The dictionary vector statistics may vary slowly temporally and hence exhibit non-stationary behavior. Neighboring vectors in a real signal signals typically have a strong correlation on each other. The correlation between neighboring vectors in a signal allows one to measure the conditional distribution of the dictionary vector of an input vector to the dictionary vector of a preceding vector in a data sequence. The conditional probability distribution of a random variable has more information than its marginal distribution. The more information we observe from a code vector, the more efficiently it can be encoded.

Using the conditional distribution to encode an input vector requires the encoder/quantizer to adapt temporally and spatially to the local statistical characteristics of the neighboring vectors of the signal to be encoded. The vector quantization scheme is hence adaptive if the codebook changes in time to match the local statistics of the input sequence. It is also important to know that feedback vector quantization schemes such as predictive vector quantization and finite-state vector quantization are not classified adaptive because they

possess the property of adapting codebook to neighboring vectors to the input vector to be encoded.

Concatenating vectors to produce a "super vector" can yield substantial improvements in coding gains at a cost of an impractical computational complexity. Adaptive vector quantization reduces this imposed the imposed complexity from concatenating vectors by exploiting statistical dependency of vectors from its surrounding vectors. Address vector quantization is a classic example of adaptive vector quantization and it is discussed in the proceeding subsection.

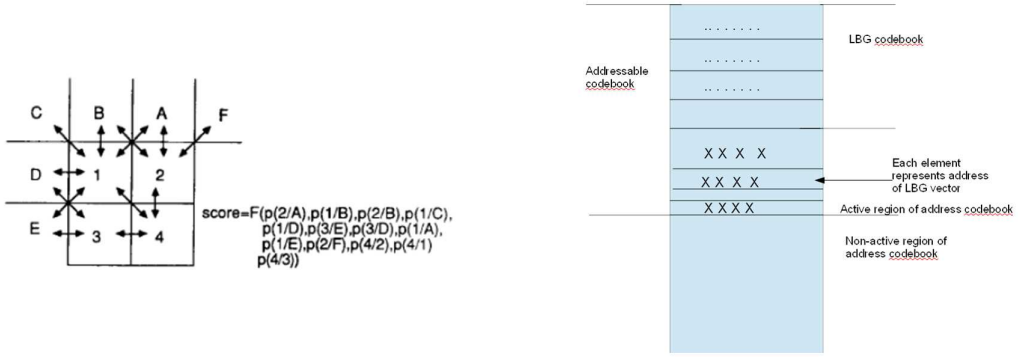
2.3.1 Address-vector Quantization

Address vector quantization is an adaptive vector quantization algorithm used primarily for image coding [31]. This quantization method uses the fact that the neighboring image blocks are correlated and follow a certain pattern. If an image block is 4×4, one can capture the correlation between four 4×4 neighboring blocks by observing that some patterns of quantization indices occur more frequently than others.

Consequently, groups of the quantization index can be encoded by an address-code book that contains the indices of image blocks that occur more frequently. The group of quantization indices for neighboring blocks in an image is called an address code vector. The address codebook hence contains a collection of all possible address code vectors. It is divided into *active* and *inactive* address codebooks. This is illustrated in figure 5.

The address-codebook is obtained by sorting the overall address code vectors according to a scoring function. The scoring function is obtained from the probability-transition matrices by multiplying (or summing) the horizontal, vertical, and diagonal probabilities between the block represented by the address code vector and neighboring blocks as shown in Figure 5.

$score = P(1/A) \times P(2/A) \times P(1/B) \times P(2/B) \times P(1/C) \times P(1/D) \times P(3/D) \times P(1/E) \times P(3/E) \times P(2/F) \times P(4/1) \times P(4/2) \times P(4/3)$



(a) (b)

Figure 5: Address vector quantization

2.4 Lossless compression schemes

The aforementioned three types of vector quantization, which are memoryless VQ, VQ with memory, and adaptive VQ, do not take into account the probability of occurrence of each of the dictionary vectors. More compression may be achieved by exploiting the non-uniformity of the distribution of the dictionary vectors in a training sequence. Lossless encoding schemes such as Huffman coding [20, 21], arithmetic coding [27] , and LZW coding schemes [4, 24, 38] capitalize on the non-uniform probability of occurrence of each of the dictionary code vectors. The following subsections briefly highlight the operation of these lossless encoding algorithms.

2.4.1 Huffman coding

Huffman coding is an efficient method of encoding a list of symbols to be transmitted when we know the probabilities of occurrence in the messages to be encoded. In this approach, the lengths of the codes assigned to the dictionary vector indices is inversely proportional to the probability of occurrence. Hence a more probable dictionary vector gets assigned a code that has smaller coding length while a less probable dictionary vector has a longer coding length. The codes assigned to every vector is developed by using the idea of a coding tree. The figure below shows the coding tree for the following example:

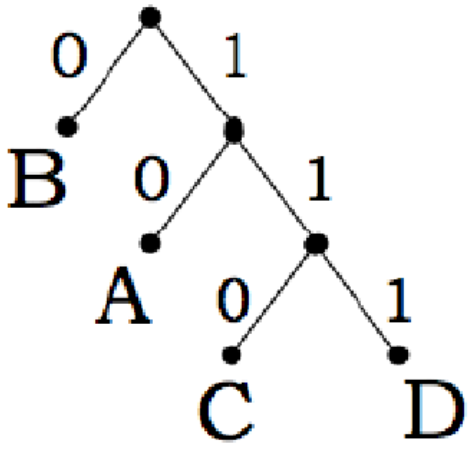


Figure 6: Huffman Coding Tree

Consider a set of symbols $S = \{A, B, C, D\}$ with probability $p = \{0.333, 0.5, 0.083, 0.083\}$.

1. Remove from S the symbols with the smallest probabilities.
2. Combine the two symbols with smallest probabilities
3. Combine the probability of the two symbols to form a new node
4. Compute the probability of new node by adding probabilities from combined symbols
5. Add new node to S where $S = \{A, B, CUD\}$, and $p = \{0.333, 0.5, 0.167\}$
6. Repeat steps 2 through 5 until S contains a single node.

The resulting is a variable-rate code for the given vector indices and probabilities. Huffman coding performs significantly better when some vectors are substantially more common than other vectors.

2.4.2 Arithmetic coding

Although Huffman's original algorithm is optimal for a symbol-by-symbol coding (i.e., a stream of unrelated symbols) with a known input probability distribution, it is not optimal when the symbol-by-symbol restriction is dropped, or when the probability mass functions are unknown. Arithmetic coding [42] is superior in many respects to the better-known Huffman method for two main reasons. First, arithmetic coding assigns fractional bits per data vector in contrast to Huffman coding which assigns whole bits. Secondly, arithmetic coding better handles the coding of data vectors with unknown a prior ensemble probabilities than Huffman coding. Arithmetic coding is a lossless coding that approaches the Shannon theoretical bound.

2.4.3 Lempel-Zev-Welsh (LZW) compression

Entropy coders such as arithmetic and Huffman coding assume that the probabilities of occurrence of dictionary vectors are known *a priori*. Huffman coding works best by first estimating the histogram of the dictionary vectors from a training data sequence. The estimation of the histogram of vectors in an a priori manner is computationally inefficient and sub-optimal in encoding groups of vectors. Since vector probabilities are not fixed as the data vector sequence is transmitted, a one-size-fits-all Huffman code does not fully capture the patterns that exist in a sequence of vectors.

The approach proposed by Lempel, Ziv, and Welsh in [32] addresses the variable nature of the histogram of dictionary vector indices. Typical variable length coders such as Huffman coding and arithmetic coding assign a unique codebook index for every dictionary vector. The one-to-one mappring between the input vector and the codebook index yields significant reductions in coding rate. However, most signals have occurences of two or more successive code vectors being transmitted through a communication channel. For example, if a dictionary code vector $\{c_1, c_2, c_3, c_4\}$ with sequence $\{c_1, c_2, c_1, c_2, c_3, c_4, c_2, c_1, c_2\}$, we can observe that $\{c_1, c_2\}$ occurs three times in the training of channel symbols. If we

include the two symbols c_1 and c_2 as one entity in the encoding table, then more copmression can be achieved. The LZW algorithm precisely uses this technique of creating a the dictionary codebook that evolves instantaneously as opposed to the entropy coding such as arithmetic and Huffman codes which are pre-assigned. This approach overall is very useful when the initial dictionary alphabet has a small size such as that of the English alphabet. Using LZW on larger codebook sizes such as 256 or more becomes intractable as there are many permutations and combinations of dictionary vectors that could assigned to a given data sequence. Hence though LZW proposes better compression for smaller dictionary sizes, the computational complexity of assigning dictionaries for a given data sequence becomes very large.

2.5 Entropy-constrained vector quantization

In memoryless vector quantization, each data vector is encoded by a binary vector of fixed code length. For example, a data vector stream to be encoded by a codebook designed by memoryless VQ is assigned 32 bits per vector. Many times, an index stream is appended by the entropy coder to achieve more compression. One approach to combine both lossless entropy coding and memoryless VQ in one step is to minimize average distortion for a given entropy. Philip Chou in [9] introduced the *entropy-constrained vector quantization* designed a codebook based on the joint criteria of distortion and entropy.

In entropy-constrained vector quantization, the optimal code vector for a given vector is assigned by finding the input vector that minimizes the modified functional:

$$d = E(\rho(X, \hat{X}) + \lambda H(\hat{X})) \tag{1}$$

where $H(\hat{X})$ is the entropy of the VQ output and λ can be viewed as a Lagrange multiplier. If we let $i(\hat{x})$ denote the index of the VQ reproduction vector $\hat{x}$, then $H(\hat{x}) = H(i(\hat{X}))$ since the index and the reproduction vector exactly determine each other. $\rho(X, \hat{X})$ is the distortion between the input vector X and the encoded input vector $\hat{X}$.

2.6 Summary of vector quantization schemes

Vector quantization achieves compression by encoding groups of data samples at a time. Rate-distortion theory [43] propounds that as the dimension of a vector approaches infinity, the distortion for a given rate approaches a certain lower bound. Though memoryless VQ achieves better compression than scalar quantization, it does not utilize the vector-to-vector dependency in a data sequence.

Vector quantization with memory such as predictive vector quantization and finite-state vector quantization address this vector-to-vector dependency by encoding a given input vector based on the input vector that was previously transmitted in a given data vector sequence. Vector quantization with memory still however has a fixed codebook size. Adaptive vector quantization adapts the size of the codebook as the vectors are transmitted. Memoryless VQ, VQ with memory, and adaptive VQ assign indices to each of their encoded vectors. The coding rate can be reduced significantly if these indices are encoded losslessly such that there is no additional distortion to the data sequence. Lossless compression schemes simply are variable-length compression algorithms that assign codes to vectors according their probability of occurrence.

Entropy constrained vector quantization encodes an input vector by finding the code vector that minimizes the average distortion between the vector and its reconstructed version for a given entropy rate. This is unlike memoryless VQ which minimizes distortion over a fixed coding bit rate. It has been shown in several literature [14, 26] that ordering a dictionary codebook yields more compression without any addition in distortion when appended by lossless entropy schemes. In today's world of communication, there is a demand to transmit huge amounts of data at smaller rates. The computational power available today overrides the additional computational cost that appending a lossless encoder may introduce to the compression algorithms that utilize reordering. No absolute ordering scheme exists between code vectors of a signal. In the past, the ordering between code vectors was done by assigning indices to vectors according to a distance measure. This is explained in

Chapter 3 while our major contribution is introduced in Chapter 4.

CHAPTER III

CODEBOOK REORDERING FOR VECTOR QUANTIZATION

3.1 Formulation of the concept of codebook reordering for vector quantization

Every communication system consists of an encoder [9], a channel, and a decoder. The encoder is composed of a sequence of one or more lossy and/or lossless encoders. A lossless encoder can be conceptualized as an injective function mapping from source element into an encoded element. Because the function is injective, the signal may be decoded from the encoded element using the inverse function, which exists by virtue of injectivity. Although the target space must be at least as large as the source space, lossy encoders can still allow compression by making use of low entropy in a system, as is the case in entropy coders. In contrast, a lossy encoder can also be conceptualized as a function, but in this case the function is not injective - *i.e.* any number of source elements map to a single encoded element. To form a decoder, typically a mapping, based on the preimage of the function, is constructed which maps from each encoded element to a unique representative estimated source element.

For clarity, sets are denoted using calligraphic capitals and random vectors and random variables are denoted by capitalized, italicized letters.

For this particular case, consider a vector space $\mathcal{E} \subset \mathbb{R}^n$ which contains the observed random vectors indexed on t (which may represent temporal, spatial, or spatiotemporal indexing). The random vectors are represented as $X[t] = (X_1[t], X_2[t], \ldots, X_n[t])^T$. Values of $X[t]$ are mapped to an index set $\mathcal{I} \subset \mathbb{N}$ by a (non-injective) function $I[t] = \alpha(X[t]) : \mathcal{E} \to \mathcal{I}$. Each of these indices $I[t] \in \mathcal{I}$ are mapped by a bijection to a set of representative codebook vectors $\mathcal{C} \subset \mathcal{E} \subset \mathbb{R}^n$, which will be denoted $C[t] = \hat{\alpha}(I[t]) : \mathcal{I} \to \mathcal{C}$. The set

$\mathcal{C}$ is assumed to have a cardinality of M with dictionary values $C = \{c_1, c_2, \ldots, c_M\}$. The indices are remapped by a bijection $J[t] = \beta(I[t]) : \mathcal{I} \to \mathcal{I}$. Finally, a lossless entropy coder is used to map the reordered indices to a variable-length bitstream, $B[t] = \gamma(J[t]) : \mathcal{I} \to \mathcal{B}^*$ where $\mathcal{B}^*$ represents the set composed of a sequence of bits of variable length. Examples of possible entropy coders include the Huffman coder or the arithmetic coder.

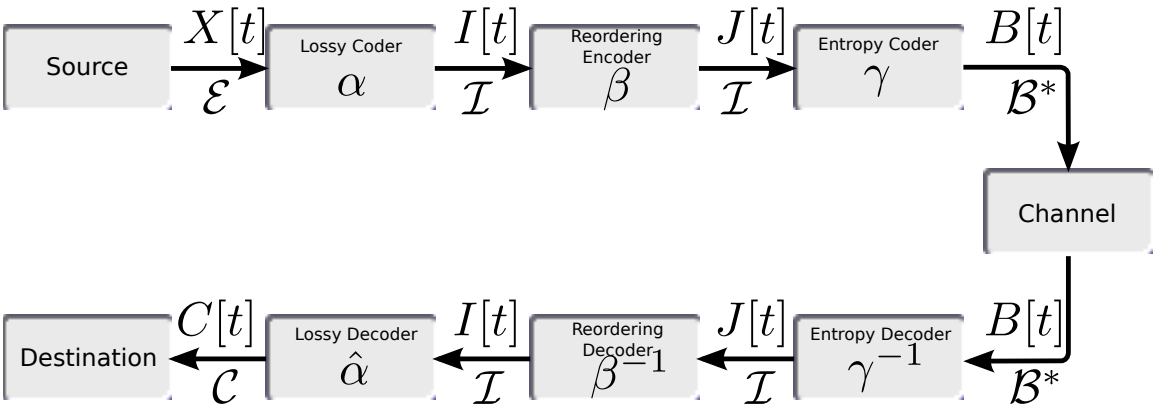


Figure 7: Variable rate communication system

Using this framework, the complete encoder can be considered as the composition of functions $B[t] = \gamma \circ \beta \circ \alpha\,(X[t]) : \mathcal{E} \to \mathcal{B}^*$ and the decoder is taken to be a similar composition $C_i[t] = \hat{\alpha} \circ \beta^{-1} \circ \gamma^{-1}\,(B[t]) : \mathcal{B}^* \to \mathcal{C} \subset \mathcal{E}$. In the adaptive system, it is assumed that the mappings are able to change in t based on source information. Changing functions are identified by a subscript t such as β_t or γ_t.

Typically, vector quantization aims to spread code vectors within the subspace inhabited by the training data. Thus, vector quantization typically results in a system where each codebook entry is approximately equally likely; this results in a uniform distribution, giving a maximal entropy.

3.2 Codebook post-processing techniques for vector quantization

In memoryless vector quantization, every input vector is mapped to one reproduction vector as explained in the previous section. The mapping between between the input vector and

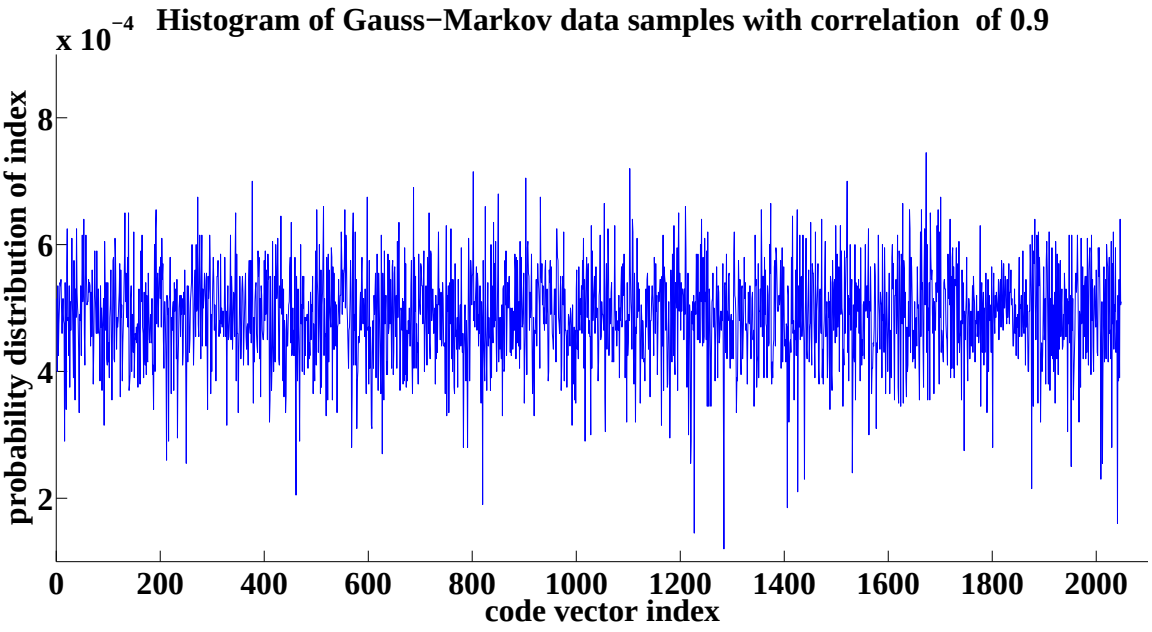


Figure 8: Histogram of VQ indices obtained from Gauss-Markov data sequence

the reproduction vector is hence an injection. At data sequence is therefore mapped into a sequence of reproduction vectors such that reproduction vector is assigned a corresponding index. Most of the time, it is this index that is transmitted through communication to the decoder to reconstruct the signal at the receiver end. This memoryless approach is good but does not vector fully exploit the inter-vector correlation in a data sequence to yield smaller data rates. Predictive VQ (PVQ) and finite-state vector quantization (FSVQ) exploit the inter-vector redundancy. In predictive vector quantization, the error vector instead of input vector is quantized. Predictive VQ exploits linear dependency among neighboring vectors while FSVQ approximates the nonlinear correlation. Both FSVQ and PVQ show improvements in bits over memoryless VQ.

In FSVQ, a vector is quantized into one of many states with each state having its unique codebook. FSVQ makes use of the next-state function to determine the state of an input vector given the preceding input vector in data sequence. This determination of the next-state function consumes a lot of computational power which can be reduced by properly choosing a next-state function. The sub-codebook is generated by a next-state function that

uses the local statistics of the previously encoded blocks to dynamically select a number of code vector from the super codebook. Though the next-state function encodes vectors based on the preceding vectors, it does not not exploit the relationship between code vectors. Dynamic codebook reordering vector quantization is discussed in the the following subsection and addresses the issue of exploiting correlation between code vectors of an input sequence.

3.2.1 Dynamic codebook reordering vector quantization

In the previous chapter, the need to exploit the non-uniformity of transition between successive vectors was introduced. In [14, 26], dynamic codebook re-ordering VQ (DCRVQ) is introduced whereby the unequal distances between vectors in a vector dictionary is utilized to produce an index stream that has smaller entropy rate. This aforementioned DCRVQ method requires only knowledge of the dictionary and nothing else. We hypothesize that more a-priori knowledge of a given input sequence can lead to more compression. While DCRVQ uses only the vector dictionary, our likelihood codebook re-ordering approach uses both the code vector dictionary and the transition probability matrix between successive dictionary vectors. This dissertation proposes a method that will yield smaller entropy rate between successive input vectors at a slightly increased computational complexity.

Suppose that $\vec{X_{seq}}$ is a sequence of vectors where $\vec{X_{seq}} = \{x_1, x_2, x_3, \ldots, x_N\}$ and $x_k \in \mathbb{R}^d$ where $k \in \{0, 1, 2, \ldots, N\}$, N is the length of vector sequence $\vec{X_{seq}}$, and d is the dimension of the vector x_k. The vector x_k is encoded into one of the dictionary vectors $s_k \in C = \{c_1, c_2, c_3, \ldots, c_M\}$ where M is the size of the codebook. Thus a data sequence $\vec{X}$ is encoded into a sequence $S = \{s_1, s_2, s_3, \ldots, s_N\}$ where $s_k \in C$ as defined in Section 3.1. If we were to treat the transition between states as a random variable, then it can be easily observed that this transition exhibits a non-uniform probability distribution function. In other words, $p(s_i|s_{i-1})$ has a non-uniform probability distribution where s_i is the state of the current vector, and s_{i-1} is the state of the previously transmitted vector in a sequence.

Dynamic codebook re-ordering (DCR) [13] vector quantization captures correlation by transmitting the index as an order of the distance between successive input vectors, while the likelihood codebook re-ordering (LCR) vector quantization uses transition probability to compute the order of the probability of occurence between successive input vectors. The transition probability matrix offers information about the correlation between successive input vectors. In our algorithm, the encoded input vector is transmitted in the order of decreasing transition probability from the previously encoded input vector. Before introducing the likelihood codebook re-ordering vector quantizer, the concepts of entropy and rate-distortion are introduced.

One of the fundamental results of the theory of coding developed by Claude Shannon is that coding systems perform significantly better if symbols are operated on as a set rather than as individual symbols. Let $\vec{\mathbf{x}}$ be an N dimensional vector of samples or some discrete features extracted from a signal. In chapter 2, the input vector $\vec{\mathbf{x}}$ is quantized into one of K pre-determined code vectors. The prototype vectors are designed by the Lloyd algorithm [29] that seeks to minimize overall distortion in system.

Let $\vec{x}$ be quantized to one of K pre-determined code vectors that are stored in a codebook C. Let the codebook C contain the code vector c_k where the index $k \in \{0, 1, 2, \ldots, K-1\}$ indicates the location of c_k in C and M is the codebook size. If we assume that the input vector $\vec{x}$ is a random variable in a d-dimensional vector space V and let ζ be a set of symbols. The vector quantization encoder can be defined as:

$$Q : \vec{x} \rightarrow c_i \text{ where } Q \text{ maps } \vec{x} \text{ into a code vector } c_i \text{ where } i \in \zeta.$$

In traditional LBG VQ, the symbol i is selected to be the index k of c_k that minimizes a distortion measure $d(\vec{x}; c_k)$. Thus, in a traditional VQ system, $i = k$. The VQ decoder reconstructs $\vec{x}$ using the code vector c_k. The VQ of $\vec{x}$ to c_k can be represented as a function Q, $Q(\vec{x}) = c_k$ and the output of the encoder for the input $\vec{x}$ is k. If input vector derived from real signals are considered as random variables, then we can associate a probability

$p(i)$ with every $i \in \zeta$. Let p denote the probability mass function of the symbols, then the entropy of these symbols for a given signal source [8] is

$$H = -\sum_{i} p(i) \log_2(p(i]))$$

For a well-designed vector quantizer, each code vector is equally likely to be chosen; that is, a long-term histogram of the codebook indices generated from quantization process is likely to be flat [23]. This maximizes the entropy of indices from quantized data frames. Real-world signals such as speech, audio, and images have strong temporarily and spatially correlated regions. Such strong correlations can be exploited such that adjacent code vectors that are *close* to each other according some cost function may be assigned similar or same symbols in the re-ordered index space.

3.3 Rate-distortion theory

A major goal in data compression is to minimize the bit rate for a desired level of distortion. It is necessary to know the theoretical lower bound on the bit rate of any quantizer. The knowledge of this bound enables one to design more complex quantizers that approach this bound. Rate-distortion theory deals with this issue of obtaining lower bounds.

For a given distortion D, one can compute the rate-distortion function or one can conversely compute the distortion for a given bit rate. The latter is called a distortion-rate function D(R). Rate-distortion theory applies to all forms of source coding including VQ.

In general, L vectors are coded by $\log_2(L)$ bits per vector. This number represents the upper bound of a VQ scheme. The minimum average rate to code the vectors is given by the entropy H(y) defined as

$$H(y) = -\sum_{i=1}^{L} p(y_i) \log_2 p(y_i)$$

$H(y)$ is the entropy of the discrete-amplitude variable y and $p(y)$ is the discrete probability. Each vector is coded by $B_i = -\log_2 p(y_i)$ bits so that vectors with different probabilities are assigned different code lengths. This results in variable-length coding with an average bit rate equal to $H(y)$.

Supposing an N-dimensional input vector $\vec{\mathbf{x}}$ is quantized to $\vec{\mathbf{y}}$ where $y = q(x)$ and $y \in Y = \{y_i, 1 \leq i \leq L\}$. The average distortion between y and x is given by $E(d(x, y))$ where $d(x, y)$ is the distortion per dimension. The vectors can be transmitted at an average bit rate of $R = H(y)/N$ bits per sample. The minimum achievable distortion $D_N(R)$ for a given rate is given by

$$D_N(R) = min_{q(x)} E(d(x, y)) \text{ with } H(y)/N \leq R.$$

As N approaches infinity, $D_N(R)$ approaches $D(R)$ where $D(R)$ is the minimum attainable distortion in coding the source $x(n)$ as rate R approaches the lower bound. The implication of the prior statement is that coding performance is enhanced as vector dimension increases.

3.4 Conclusion

In this chapter, we discuss the concept of codebook reordering and its origins. We observe that codebook reordering yields reduction in coding rate with a minimal increase in computational complexity. The effectiveness of the codebook reordering procedure arises from the fact that the entropy reduction is achieved without any increase in distortion as compared to a standard VQ system. While incorporating the DCR in the encoder and the decoder of the VQ will result in an increase in the complexity, some of it can be alleviated by pre-calculating the reordering or by performing a partial reordering.

CHAPTER IV

LIKELIHOOD CODEBOOK REORDERING FOR VECTOR QUANTIZATION

4.1 Exploiting the transition probabilities between vectors

In the previous chapter, the opportunity to exploit the non-uniformity of transition between successive vectors was introduced. In [14, 26], dynamic codebook re-ordering VQ (DCRVQ) was introduced whereby the unequal distances between vectors in a vector dictionary is utilized to produce an index stream that has smaller entropy rate. The DCRVQ method requires only knowledge of the dictionary and nothing else. This dissertation proposes a method that will yield smaller entropy rate between successive input vectors at a slightly increased computational complexity. We hypothesize that more *a-priori* knowledge of a given input sequence can lead to more compression. While DCRVQ uses only the vector dictionary, our likelihood codebook re-ordering approach uses both the code vector dictionary and a matrix of transition probabilities between successive dictionary vectors.

Suppose that $\vec{X}$ is a sequence of vectors where $\vec{X_{seq}} = \{x_1, x_2, x_3, \ldots, x_N\}$ and $x_k \in \mathbb{R}^d$ where $k \in \{0, 1, 2, \ldots, N\}$, N is the length of vector sequence $\vec{X_{seq}}$, and n is the dimension of the vector x_k. The vector x_k is encoded into one of the dictionary vectors $c_k \in C = \{c_1, c_2, c_3, \ldots, c_M\}$ where M is the size of the codebook. Thus a data sequence $\vec{X}$ is encoded into a sequence $S = \{s_1, s_2, s_3, \ldots, s_N\}$ where $s_k \in C$ as defined in Section 3.1. If we were to treat the transition between states as a random variable, then it can be easily observed that this transition exhibits a non-uniform probability distribution function for a correlated source. In other words, $p(s_i|s_{i-1})$ has a non-uniform probability distribution where s_i is the state of the current vector, and s_{i-1} is the state of the previously transmitted vector in a sequence.

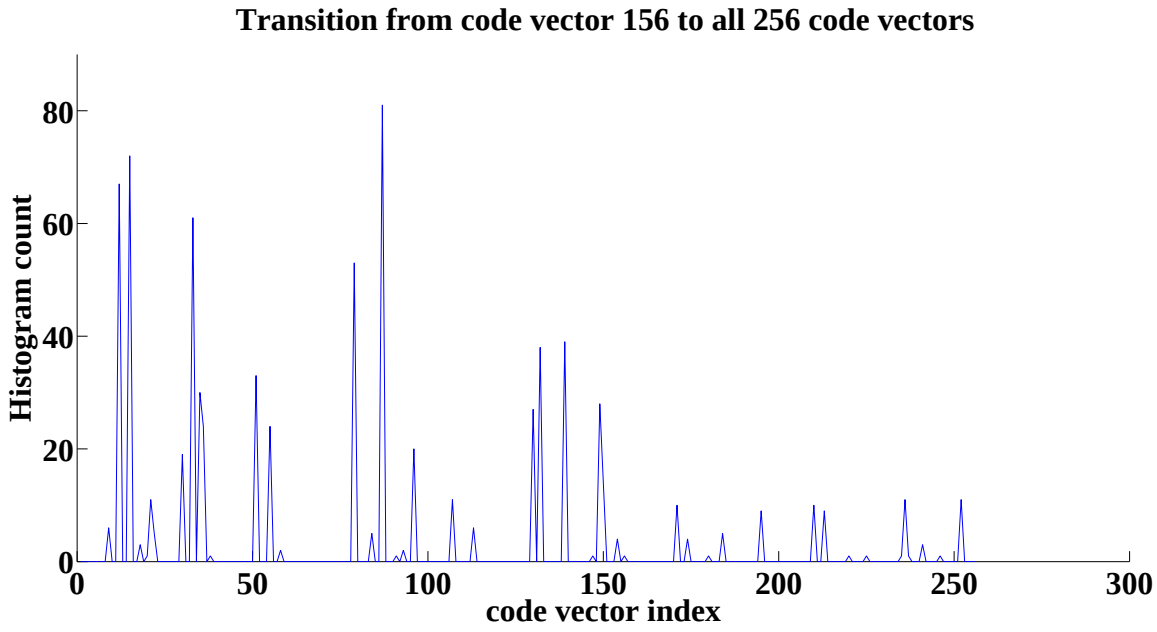


Figure 9: Plot of transition counts between between code vector 156 and other 255 vectors

Figure 9 demonstrates the non-uniformity of the number of transitions between one code vector and another for a given codebook. This figure is obtained by vector quantizing a Gauss-Markov correlated sequence with a 256 code vector codebook and noting the frequency of the vectors that follows vector 156. It does not display approximate uniformity. This non-uniformity of the probability distribution is beneficial especially when the transitions are encoded with entropy coding. The transition matrix is the count of the transitions between all possible combination of successive transitions between successive dictionary vectors. For example, if there are 8 dictionary vectors, then the transition matrix will be an 8 x 8 matrix of transition counts. The transition probability matrix is obtained by normalizing the the transition matrix obtained from the a given input vector sequence.

Consider the encoded vectors as described below: $S = \{c_2, c_2, c_1, c_4, c_1, c_1, c_2, c_2, c_2, c_2, c_2, c_2, c_2, c_2, c_1$ the transition matrix is given below in table 2. When the transition probabilities are sorted, the dictionary vector transitioned to with the highest transition probability is assigned 0; the vector with the 2^{nd} highest probability is assigned index 1, and the vector with the n^{th} highest probability is assigned index $n - 1$.

Table 2: Example 4×4 transition matrix for 30-element training sequence

	c_1	c_2	c_3	c_4
c_1	1	1	1	3
c_2	4	11	0	0
c_3	0	2	0	2
c_4	1	0	3	0

4.2 Likelihood codebook re-ordering vector quantization (LCRVQ)

The vector-quantized estimates of successive input vectors in a correlated signal typically produces reproduction vectors that are closer to each other in the L_2 distance sense. This property can be exploited by assigning new indices to each vector in the dictionary after a vector is encoded; these indices are assigned, beginning at 0 and increasing in order of their closeness (in an L_2 sense) to the previously used dictionary vector. Since correlated signals have successive dictionary vectors that are closer to each other than an ordinary Gaussian signal, correlated signals will have more new indices with values closer to 0 compared to an uncorrelated signal. This aligns the distribution of the new indices closer to zero such that the indices closer to 0 occur more frequently. This approach is called reordering vector quantization because the vectors are assigned new indices according to previously transmitted vector.

The logic behind the likelihood codebook reordering vector quantization is that some transitions between code vectors are more likely to occur than other transition. For a given vector X_i, the proceeding vector X_{i+1} is assigned indices closer to 0 when $Q(X_{i+1})$ more frequently follows $Q(X_i)$. In this case, X_i is the vector at time i and $Q(X_a)$ is the code vector of a vector X_a. After the reordering takes place, it is typically observed that the conditional probability histogram of occurrence of the reordered index given a previously transmitted signal is monotonically decreasing. The monotonic decay can benefit from further lossless coding such as arithmetic coding and Huffman coding and hence result in reduction in coding bit rate. The bit rate can be further reduced if we consider that for a given vector, some code vectors are less likely to follow others in a training sequence.

In other words, the number of transitions for these "less likely" code vectors is smaller than for the other dictionary vectors. The non-uniformity of transition of the code vectors implies that the image can be encoded by a fractions of the codebook at the cost of additional to the signal. For example, if we consider a codebook of size 8 given as $C = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8\}$ where the vector X_i is transmitted as code vector c_2, and the following inequality is observed:

$p(c_7|c_2) \geq p(c_5|c_2) \geq p(c_3|c_2) \geq p(c_2|c_2) \geq p(c_4|c_2) \geq p(c_1|c_2) \geq p(c_6|c_2) \geq p(c_0|c_2)$.

For the inequality above the reordered codebook for vector X_i previously transmitted as c_2 is $\{c_7, c_5, c_3, c_2, c_4, c_1, c_6, c_0\}$. However, we may want to encode the data sequence using half of the codebook to improve coding. The most reasonable approach will be to encode the vector X_{i+1} using the four most likely code vectors $\{c_7, c_5, c_3, c_2\}$ and discard the last four $\{c_4, c_1, c_6, c_0\}$. The process of selecting the L^{th} most frequent vector from the reordered codebook is called truncation. This truncation consequently yields a much more reduced bit rate for a slight increase in distortion. This distortion is negligible as the bulk of the input signal is captured in the retained portion of the re-ordered dictionary codebook. Our approach of truncation is one way of performing finite-state vector quantization. In our approach, sub-codebook selection is coupled with reordering.

Let the matrix, Q, be the transition matrix that is composed of all possible transitions, t_{ij} from code vector c_i to code vector c_j. To obtain this transition matrix, the input data sequence is first encoded by the LBG vector quantization to obtain a sequence of dictionary vectors. The transition probability matrix, P, is the matrix of the probability of transition from one code vector to another. This matrix, P, is obtained by normalizing each row of the transition matrix, Q. Equation 4 explains the preceding statement. To illustrate this, consider a training data sequence encoded by the following dictionary vector sequence. Consider a 2-bit codebook $C = \{c_1, c_2, c_3, c_4\}$ that encodes a training data sequence to give the following sequence:

$$S = \{c_1, c_2, c_3, c_1, c_2, c_4, c_4, c_1, c_2, c_3, c_2, c_3, c_3, c_4, c_1, c_4, \\ c_2, c_3, c_3, c_4, c_1, c_3, c_3, c_4, c_1, c_3, c_2, c_2, c_4, c_4, c_1, c_1\} \tag{2}$$

The training data consists of 32 vectors. The transition matrix is estimated from this sequence of code vectors is shown below:

	c_1	c_2	c_3	c_4
c_1	1	3	2	1
c_2	0	1	4	2
c_3	1	2	3	3
c_4	5	1	0	2

To estimate the transition probability from c_1 to c_2, P_{12}

$$p(c_1|c_2) = \frac{q(c_1|c_2)}{q(c_1|c_1) + q(c_1|c_2) + q(c_1|c_3) + q(c_1|c_4)} \tag{3}$$

To transition from c_i to c_j, P_{ij}, we calculate

$$P_{ij} = p(c_i|c_j) = \frac{q(c_i|c_j)}{\sum_{k=1}^{N} q(c_i|c_k)} = \frac{Q_{ij}}{\sum_{j=1}^{L} Q_{ij}} \tag{4}$$

where $q(c_m|c_n)$ is the count of the transition from code vector c_m to code vector c_n.

When quantizing a source, each vector is assigned an index based on the transition likelihood from the previously used code vector. The most likely code vector is assigned 0 and the next most likely vector is assigned index 1, and so forth. The result is that, according to the transition likelihood, indices closer to zero will be more common, making the index stream more amenable to further entropy coding such as Arithmetic or Huffman coding.

4.3 LCR vector quantization algorithm

The input data is assumed to be a set of ordered vectors $\{x_t \in \Re^n\}$ originating from some source $\mathbf{S}$ where t is a discrete-time index. In other words, the input data $\vec{X_{seq}}$ can be

expressed as $\{x_1, x_2, x_3, \ldots, x_L\}$. For every time instant, t, the vector x_t is mapped to one of the indices in the index set I_{abs} where $I_{abs} = \{i : i = 1, 2, 3, \ldots, L\}$ and L is the number of vectors in dictionary codebook. The index train for the input data is given as $I = \{I_0, I_1, I_2, I_3, \ldots, I_N\}$ where I_t is the index of the dictionary vector that closest to vector x_t.

The dictionary codebook, $\mathbf{C} = \{c_1, \ldots, c_N\}$, is generated using the LBG algorithm using the algorithm described in [29] from the training data set. These training vectors are then used to estimate the transition matrix P. Each element of P, that is, P_{ij}, represents the likelihood of the code vector c_j following the code vector c_i, or in other words, the likelihood of an input vector, x_{t-1}, that is quantized to code vector i being followed by an input vector, x_t, that is quantized to code book vector j.

Table 3: Transition Matrix and Reordered Codebook

	Likelihood of $c_i \vert c_{prev}$			
c_{prev}	c_0	c_1	c_2	c_3
c_0	0.15	0.17	0.33	0.35
c_1	0.10	0.20	0.40	0.30
c_2	0.35	0.25	0.10	0.30
c_3	0.10	0.30	0.20	0.40
	Re-ordered indices			
c_{prev}	c_0	c_1	c_2	c_3
c_0	3	2	1	0
c_1	3	2	0	1
c_2	0	2	3	1
c_3	3	1	2	0

Let p represent the previously transmitted index, and m the re-ordered index to be transmitted for the current input vector. Note that different variables other than i and j are used to emphasize the fact that the transmitted or stored index is not the same as the absolute index of the code vector. The reordering function, $\varphi(i, j)$, has inputs i, and j. Therefore, the transmitted index gets assigned the value obtained from the re-ordering function.

4.3.1 VQ encoder with LCR

The LCR encoder is described as follows:

1. We always assign the absolute index at the discrete-time, $t = 0$, to the re-ordered index. In other words, $m(t = 0) = I_0 = i$

2. Assign j = the previous i. For each discrete-time instant, t, in the training set, quantize the vector, X_t, and note the corresponding absolute code vector index, i.

3. The transmitted index is $m(t) = \varphi(i, j)$

4.3.2 VQ decoder with LCR

After the re-ordered indices passes through the channel, it is decoded by the following algorithm.

1. Absolute index at ($t = 0$) = $m(t = 0)$

2. For t=1, 2, 3 and above $\hat{I}_t = \varphi^{-1}(j, m)$ where j is the absolute index, and m is the re-ordered index, and φ^{-1} is the inverse dynamic map. The inverse dynamic maps the absolute index at time instant $t-1$ and the reordered index at time instant t to the absolute index at time instant t. Hence $\varphi^{-1} : \mathbb{Z}^2 \to \mathbb{Z}$.

3. The reconstructed vector at time t is c_l where $l = \hat{I}_t$

Tables 3 illustrate the concept of likelihood codebook re-ordering vector quantization. We assume a codebook with 4 code vectors, and a codebook transition matrix as in Table 3. Our algorithm is also presented in [10].

4.4 Truncation of LCR Vector Quantization

Suppose that the likelihood re-ordering vector quantization has been used to compress a data sequence. Empirical observation of the transition matrix will show that some vectors

are more likely to be transmitted than other vectors given any previously transmitted vector. For example, let us assume that a data sequence is quantized by four vectors $\{c_1, c_2, c_3, c_4\}$ and vector c_1 is transmitted at time, $t-1$. If the re-ordered codebook given the previous vector c_1 is $\{c_4, c_1, c_2, c_3\}$, and the vector c_2 transmitted at time, t, will normally be assigned the index 2 since it is the third most likely to be transmitted given the previous vector. However, there might be a need to represent the four-vector codebook by a two-vector codebook to avoid the need of entropy coding though at the cost of a slight increase in distortion. For this example, by encoding the vector at time t using the two most likely vectors, c_4, and c_1, one can avoid using an entropy coder.

The truncation procedure takes place after probability transition matrix is estimated. Once the transition probability matrix, P, is estimated, the dictionary vectors are sorted so that we have a re-ordered transition matrix P^1. If we consider that $P = \{P_{ij} : 1 < i < L, 1 < j < L\}$, then the re-ordered transition matrix $T^1 = \{p^1_{ij} : 1 < i < L, 1 < j < L\}$, where p^1_{ij} is the j^{th} most likely dictionary vector in transition vector $P_i = \{P_{i1}, P_{i2}, P_{i3}, \ldots, P_{iL}\}$. Therefore to truncate an L-code vector codebook by 2, we need to know the code vector previously transmitted and the re-ordered transition codebook. For example, if the code vector c_m is previously transmitted, the codebook truncated by 2 is $C_T = \{p^1_{m1}, p^1_{m2}, p^1_{m3}, \ldots, p^1_{mL/2}\}$. Consequently to encode any vector following c_m, we find the closest vector in C_T that matches the vector to be encoded.

$$p^1_i \in C$$

Generally, a codebook truncated from size L to K is based on the previously transmitted code vector and the re-ordered codebook. In this case,

$$C_T = \{p^1_{m1}, p^1_{m2}, p^1_{m3}, \ldots, p^1_{mL}\}.$$

The truncated encoded version of dictionary vector c_j given that vector c_i was previously

transmitted is determined by

$$\min_{l} \|c_j - p_{mL}^{1}\|_2.$$

Truncation of the reordered codebook is a special case of the finite-state vector quantization whereby the sub-codebook indices is selected based on the conditional probabilities of transition rather than on the current"state" of the vector being quantized. Additionally the histogram of the indices from finite-state vector quantization still tends to be uniform with respect to the skewed nature of the reordered and truncated indices. Truncation is beneficial because it captures the essence of a given signal at a much reduced coding rate without yielding much increase in expected distortion. Decreasing the codebook size by a factor of 2 decreases the rate by 1 bit per vector. Consequently, if the codebook size is decreased by 2^n we reduce the rate by n bits per vector. Thus, if we start with a large code book, truncating back to some desired rate will yield much lower distortion than using a non-truncated code book at the same rate as will be shown later.

4.5 Empirical observations of LCR algorithm on Gauss-Markov data

Likelihood codebook re-ordering is derived from dynamic codebook re-ordering. To understand its behavior properly, it is necessary to compare the performance of the LCR algorithm to the DCR algorithm. Some analyses are made on Gauss-Markov data to illustrate the relationship between the correlation between successive vectors and the coding bit rate of a given re-ordering approach. In one experiment, a 300000 by 10 Gauss-Markov data sequence is generated. The correlation is varied for this data while observing the change in entropy. In another experiment, the effect of increasing the dimension of Gauss-Markov vector on entropy rate is observed for different correlation coefficients. In both experiments, a 300000 by 10 Gauss-Markov sequence is used to generate the codebook which is subsequently used to generate the transition matrix for the testing data sequence.The

Gauss-Markov data is generated according to equation below:

$$y_t = \alpha y_{t-1} + (1 - \alpha)x_t \tag{5}$$

In our experiments, we assume the following generational model of our Gauss-Markov data as in Equation 5 where x_t is an identically independent Gaussian random variable, $\alpha \in (-1, 1)$ is the correlation parameter, and y_t is the source sample at time t. The vectors are formed by grouping y_i's into groups of 10.

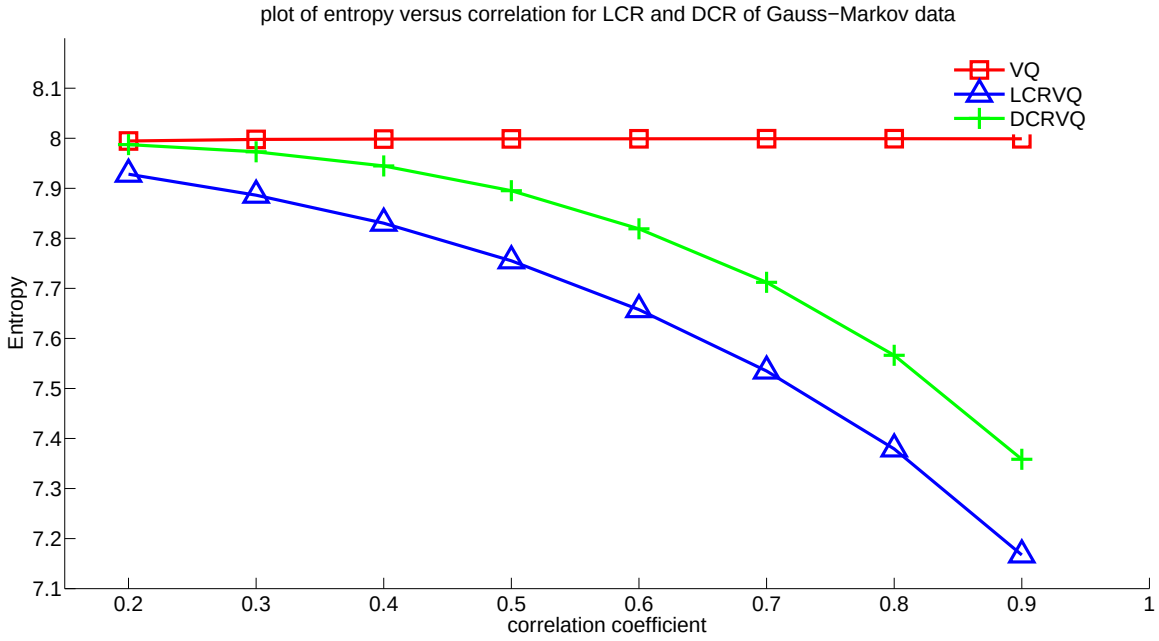


Figure 10: Entropy vs correlation of LCR and DCR algorithm

In the second experiment, we have a 300000 vector sequence with dimensions that change from 2 to 12. For both sets of experiments, we see that LCR indices have a smaller entropy than DCR indices. This is illustrated in Figure 10. Figure 11 shows that the more correlated signals produce smaller entropy coding rates than the less correlated counterparts. All these results and the results in subesequent chapters are significant because they the prove that our algorithm exploits the correlation between successive dictionary vectors better than any of the other existing methods such as finite-state vector quantization, and dynamic codebook re-ordering.

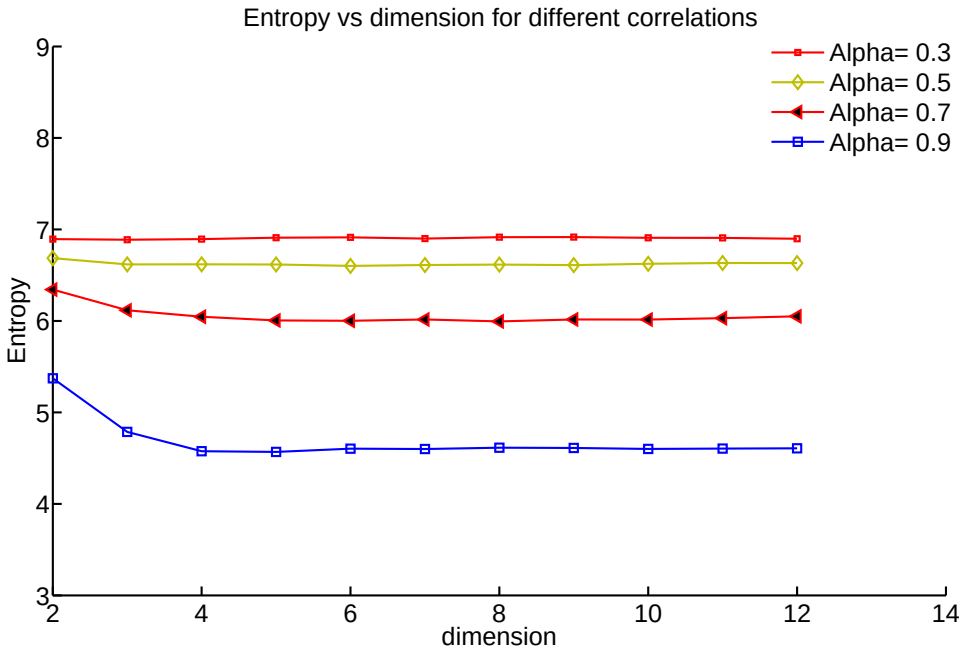


Figure 11: sample entropy vs dimension of LCR at different correlations

4.6 Computational complexity of LCRVQ

The VQ codebook is usually found using the k-means algorithm on a given data sequence. The k-means algorithm is a numerical method that calculates the centroids of a data sequence by dividing the data sequence space into Voronoi regions with each region being mapped to a centroid. The training data sequence typically is a sequence of vectors with a dimension greater than or equal to two. The two-dimensional data segmentation into Voronoi regions is depicted in the Figure 12 below.

The computational complexity of the VQ algorithm is equal to the computational complexity for the k-means algorithm. Reordering the memoryless VQ algorithm results in a marginal increase in computational complexity relative to VQ. The Table 4 below describes the comparison between codebook reordering and other cardinal vector quantization methodologies.

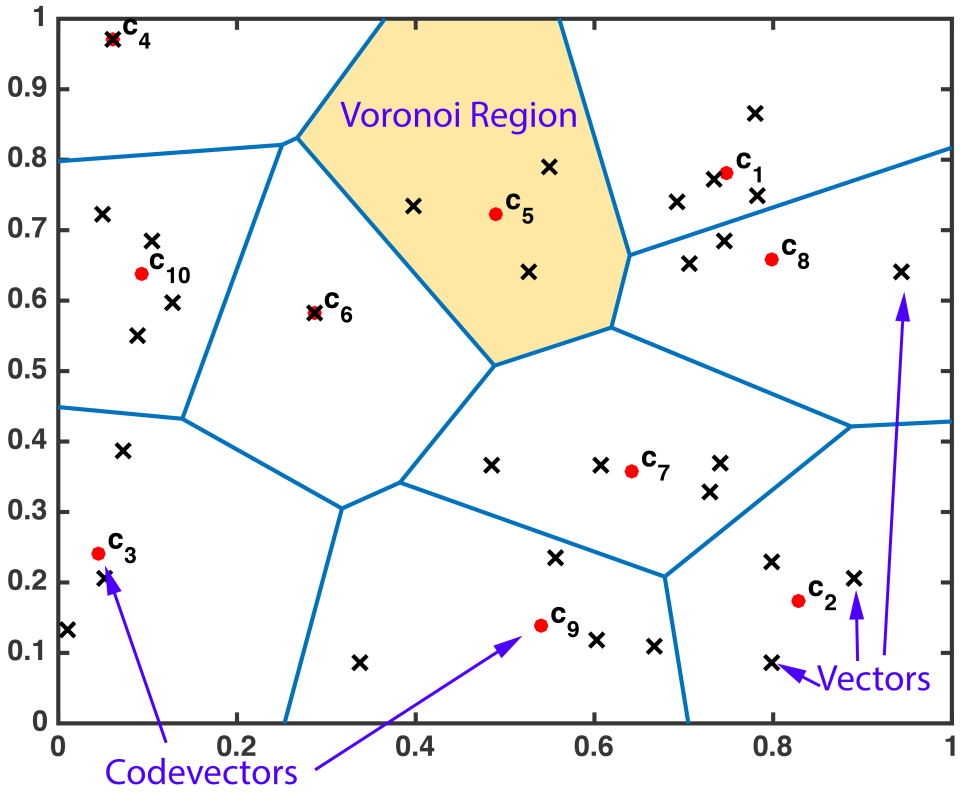


Figure 12: Voronoi region segmentation of the 2-D data space

Table 4: Voronoi regions

FSVQ	**VQ**	**VQ+LCR**	**VQ+LCR+truncation to L**
$O(iCKNd)$	$O(iKNd)$	$O(iKNd + KN + NKlogK)$	$O(iKNd + LN + LNlogL)$

where i is the number of iterations, K is the number of vectors in codebook, N is the number of data vectors, L is the number of elements to search in truncated codebook, d is the dimension of data vector.

We can see from the Table 4 that LCR with VQ has a slightly increased computational complexity compared to VQ. If we have the same number of code vectors per codebook and more than one state, the FSVQ algorithm takes up more computational complexity compared than VQ.

CHAPTER V

APPLICATION OF LIKELIHOOD CODEBOOK REORDERING ON GAUSS-MARKOV DATA

5.1 One-dimensional Gauss-Markov data model

In many practical applications of signal processing methodologies, a newly developed method is most often tested on a generic dataset to observe its qualitative performance. These datasets have properties similar to the real dataset to be tested upon. In our specific case of vector quantization, speech, image and video signals are three of the primary types of data upon which we measure coding rate performance. The underlying relation between the three types of data are the spatial and temporal correlation from sample to sample or pixel to pixel values. This correlation can be easily mimicked by generating a sequence that has a correlation from sample to sample. As described above, the Gauss-Markov data is used in this dissertation to perform a preliminary test of our proposed likelihood codebook reordering vector quantization scheme. In our experiment, the Gauss-Markov model is described as

$$y_t = \alpha y_{t-1} + (1 - \alpha)U_t \tag{6}$$

where y_t is the Gauss-Markov sample at time t, $\alpha \in (-1, 1)$ is the correlation coefficient , and U_t is an identically independent Gaussian random variable.

This model has been used extensively in testing VQ algorithms such as FSVQ and PVQ.

5.2 Experimental results on Gauss-Markov data

Unless otherwise specified, we set L to 10 and α to 0.9. We use a training dataset of M=500000 data vectors, and testing set of N=100000 data vectors. We seek particularly to

compare the rate-distortion curves of LCR, DCR, FSVQ, LBG, truncated LCR, and truncated DCR algorithms. As mentioned earlier, we also explored the result of changing the correlation parameter on the entropy of synthetic data set. The result of changing the correlation coefficient is demonstrated in Table 5.2. Table 5.2 shows that increasing correlation coefficient between vectors decreases entropy for memoryless vector quantization, vector quantization with LCR, and VQ with DCR. We also observe that LCR achieves lowest entropy for a given correlation factor. Figures 13 and 14 are the histogram and log-histogram of the re-ordered indices for both LCR and DCR algorithms. Both aforementioned figures show that the LCR algorithm has a skewed histogram. Figure 14 highlights that the data that is originally quantized by 2048 dictionary vectors can be represented effectively by only 128 vectors. This representation is about 1/16 of the original size of the codebook. The fact that the majority of the histogram is biased only towards the first 128 indices points one to the opportunity for truncation of the dictionary codebook.

Truncation of the codebook allows one to represent a data sequence with smaller distortion for a given coding rate. The concept of truncation is explained in Section 4.4. The skewed nature of the histogram also highlights that reordered indices will have a smaller entropy than the un-reordered indices. The fact that the majority of the indices close to zero have higher frequencies is an implication of the benefit that results if entropy coding such as Arithmetic coding or Huffman coding is applied. We observed in Figure 31b that Gauss-Markov data compressed by a 2048 codebook can be represented without significant increase in distortion with a 128 code vector dictionary. This approach is very powerful as it reduces entropy of a compression. When compared to FSVQ, the LCR algorithm has a higher rate-distortion performance even when the dictionary of code vectors is truncated by a factor of 1/16.

Table 5: Table of index entropies versus correlation in Synthetic data comparing LBG, DCR, and LCR algorithms. Codebook of size 16, and training data of dimension 100000 by 10

Correlation	LBG	DCR	**LCR**
0.85	2.9778	2.0430	**1.2697**
0.90	2.9663	1.8288	**1.1681**
0.95	2.9303	1.5019	**0.9880**
0.98	2.8823	1.2927	**0.8573**

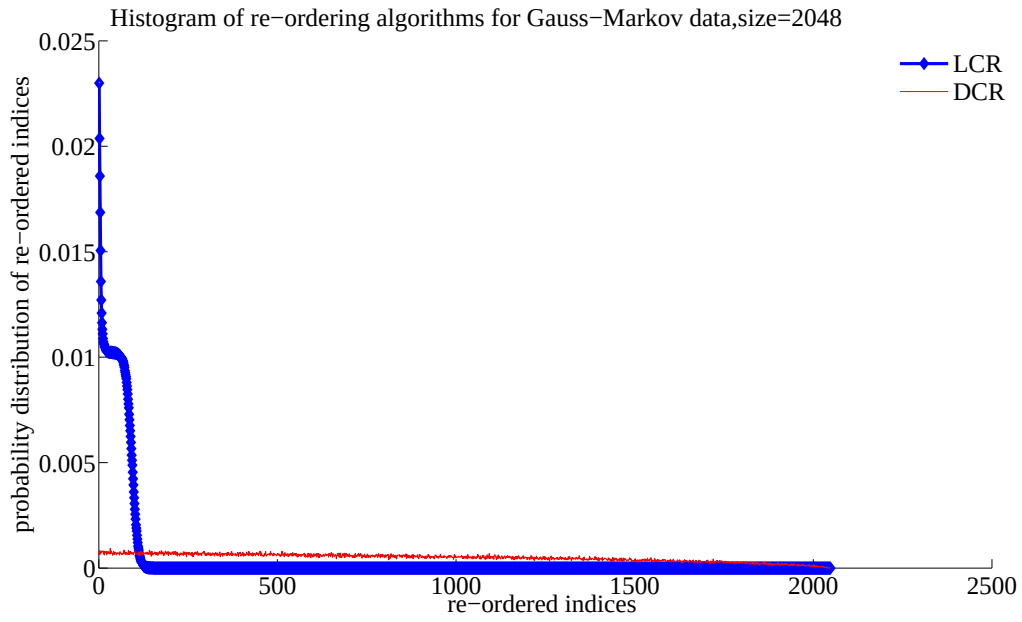


Figure 13: Histogram plot of Gauss-Markov data quantized by LCR and DCR VQ

5.3 Conclusion on the application of LCR VQ on Gauss-Markov data

Likelihood codebook reordering vector quantization results in more compressibility of a signal if the signal has higher correlation. This is shown in Table 5.2as the entropy decreases as the correlation coefficient increases. We also observe that the LCR algorithms have a better rate-distortion performance than the DCR algorithms for all levels of truncations. The results also demonstrate that the LCR has a higher rate-distortion performance than FSVQ even when the codebook of the LCR scheme is truncated up to 1/8. Our LCR methodology thus demonstrates superior performance in coding rates relative to other VQ

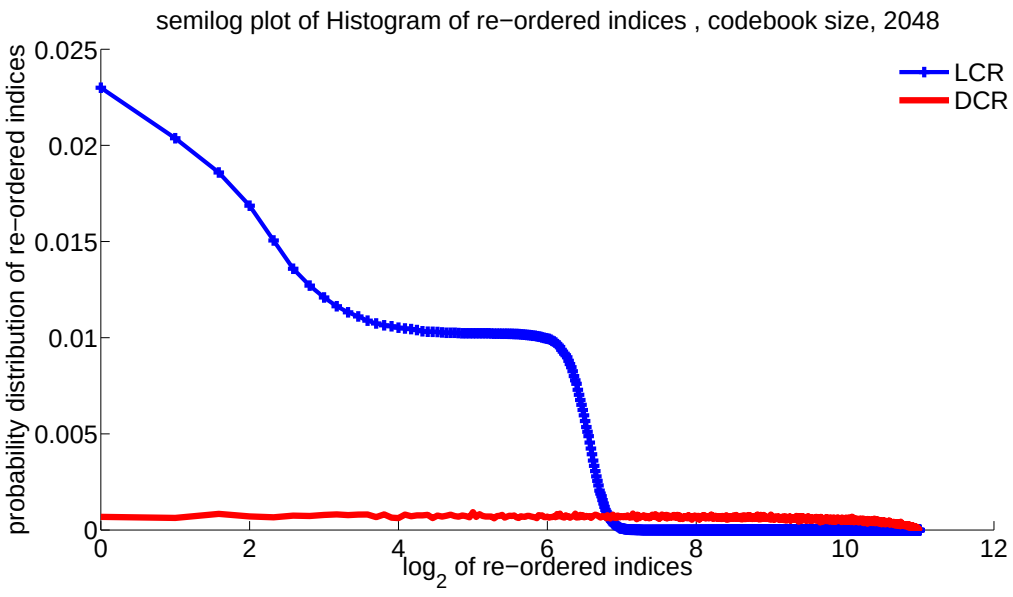
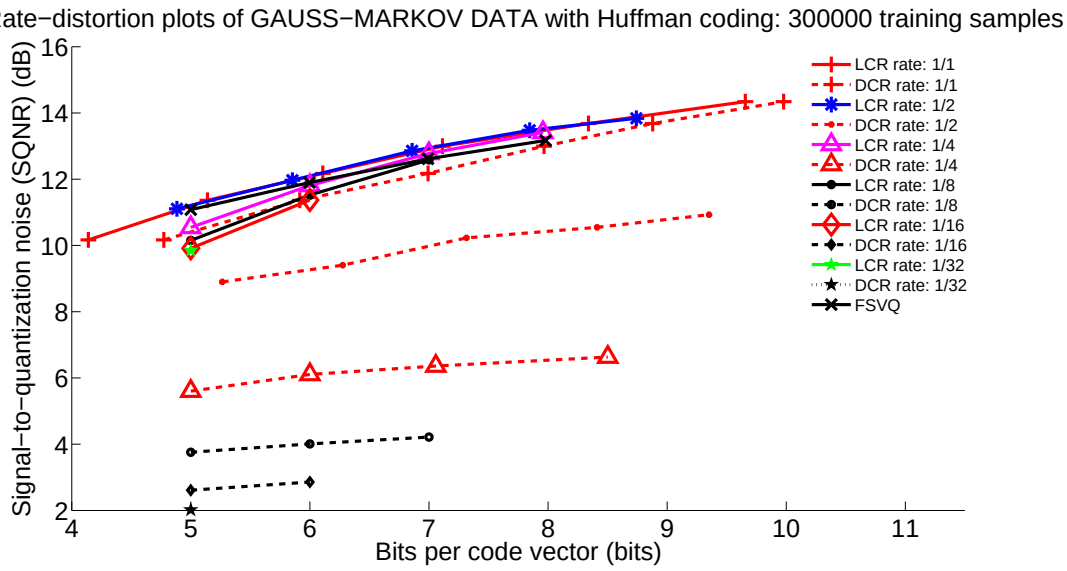
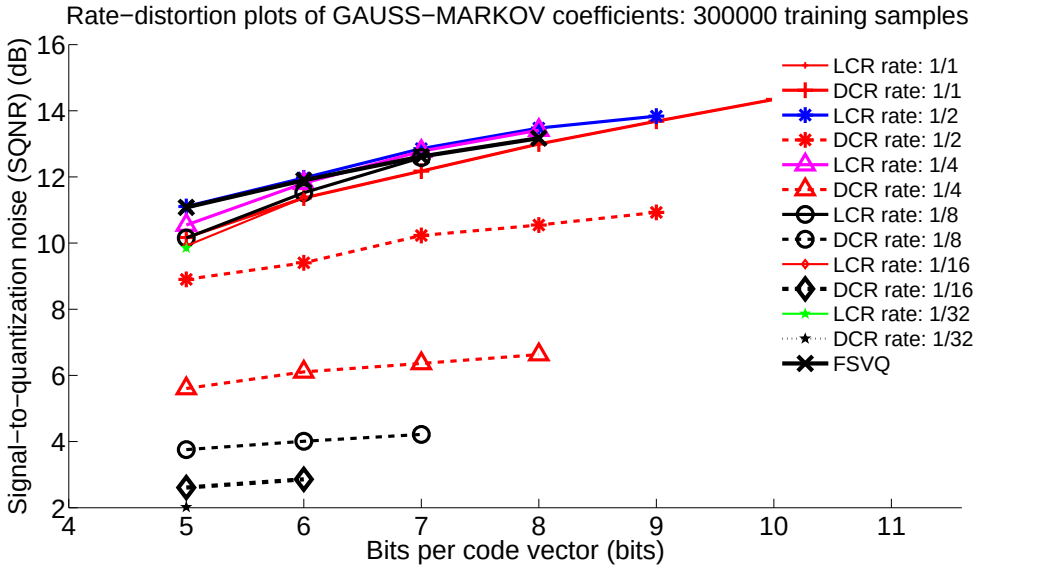


Figure 14: Semilog Histogram plot of Gauss-Markov data quantized by LCR and DCR VQ

schemes, especially when the codebook is truncated up till 1/8 for Gauss-Markov signals.



(a)



(b)

Figure 15: Rate distortion curve of Gauss Markov Data with a) Huffman b) without Huffman

CHAPTER VI

APPLICATION OF LIKELIHOOD CODEBOOK REORDERING ON SPEECH LINE SPECTRAL PAIR SIGNALS

6.1 Introduction to speech coding

Speech is a primary means of communication between humans. Due to the importance of having speech transmitted at a quality level through a communication channel using a small number of bits, it is very necessary that speech signals be compressed and transmitted with low computational cost, coding rate, and minimal distortion. These three aforementioned parameters of the computational cost, coding rate, and distortion are key indicators of how good a compression algorithm is. Speech is a one-dimensional signal and historically has high correlation between successive samples and consequently input vector samples. This mutual information between successive input vectors translates into the quantized version of these input vectors.

Speech coding can be classified into two main categories: waveform coders, and vocoders. Waveform coders strive for a facsimile reproduction of the signal waveform. In principle, they are designed to be signal-independent. They can equally code a broad spectrum of signals- speech, music, tone, voice band data, images. These coders tend to be robust to a wide variety of talker characteristics and for noisy environments. Waveform coders can be optimized and made more signal-specific for greater coding efficiency. This typically is done by observing statistics of a given signal set, so that the waveform coder yields signal coding error for a given class (speech in our case). This approach exploits statistical characteristics of speech waveform.

The second class of speech coders depends on the description of speech using a prior knowledge about the how the speech signal was generated [12, 39]. The idea is that certain

physical constraints of the signal generation can be quantified, and turned to advantage in efficiently describing the signal. This implies that the speech signal must fit into a model and characterized according to this model. The next two sections will introduce fundamentals of waveform coding and vocoding.

6.2 Coding of line spectral pairs of speech signals

Most speech coding schemes at low bit rates (Less than 8000 bits/sec) make use of the linear predictive model of speech. The parameters of this model are the linear predictive coefficients (LPC) and these contain information abou the short-term spectrum of the speech signal. A mathematically equivalent representation of the LPC coefficients is the line spectral pair (LSP) parameters. The LSP features have many interesting properties that make them more appropriate for efficient coding than LPC coefficients. The LSP parameters have an ordering property which states that the $(i+1)^{th}$ parameter is always greater than the i^{th} parameter. The different parameters within the speech frame are not independent but correlated. Hence, it is more practical and efficient to vector quantize LSP coefficients than LPC coefficients due to the robustness and lower of dynamic range relative to LPC coefficients.

6.3 Speech coders

Speech vocoders utilize the formant and harmonic structure of speech to intelligently code speech signals. In this method, the filter coefficients to model the formant and the harmonic structure of speech are coded instead of the actual waveform as in waveform coding. One of the most used application of speech vocoders is the code-excited linear prediction (CELP) algorithm [39, 40]. CELP utilizes long-term and short-term prediction models of speech synthesis to avoid the strict voiced/unvoiced classification of linear prediction coefficients(LPC). The name of "code-excited" stems from the excitation codebook, containing the "code" to "excite" the synthesis filters. The high complexity of CELP coders, was

previously thought to be impractical due to the computational complexity. Over the years, researchers have found many approaches to accelerate the encoding process making CELP a reality. CELP is used very often in speech coding and is the foundation of many standardized coders. CELP and its variant algebraic code-excited linear prediction (ACELP) are used often in speech processing algorithms because of the ability to encode speech with minimal perceptual distortion and coding rate.

6.3.1 Code-excited linear prediction

In general, a source encoder utilizes the mathematical model of the data source. The model has certain parameters and the role of the encoder is to estimate those parameters. If the original data is speech, the source coder used to estimate the unknown parameters is called the vocoder (from "vocal coder"). The vocoder is illustrated in Figure 16.

In this model, the output is the sequence of speech signals $s(n)$ coming out of the LPC filter. The input $x(n)$ to the model and filter is either a train of pulse when the sound is voiced speech or white noise when the sound is unvoiced speech. The quantities $x(n)$ are also termed innovations (a train of pulses and white noise). The task of the speech coder is to input samples $s(n)$ of actual speech, use the LPC to determine the equivalent sequence of innovations $x(n)$ and output the innovation in compressed form. The main mathematical model of the LPC estimation is given by the equation below:

$$s(n) = \sum_{i=1}^{L} a_i s(n-i) + x(n) \tag{7}$$

Each speech sample $s(n)$ is represented as a linear combination of the previous L samples plus the innovation signal $x(n)$. The weighting coefficients $a_1, a_2, a \ldots, a_L$ are linear prediction coefficients (LPC). The samples of the input speech are divided into blocks of N samples called frames. Each frame is typically 10-20 ms long (that is N = 80-160). Each frame is then divided into sub-frames of k samples (equal to the dimension of the VQ). The LPCs are typically computed using the Levinson-Durbin algorithm [16, 35]. Vector

quantization is and be used to encode these LPC coefficients in the vocoder model instead of scalar quantization. In this dissertation, we do not use vector quantization to encode the innovation signals.

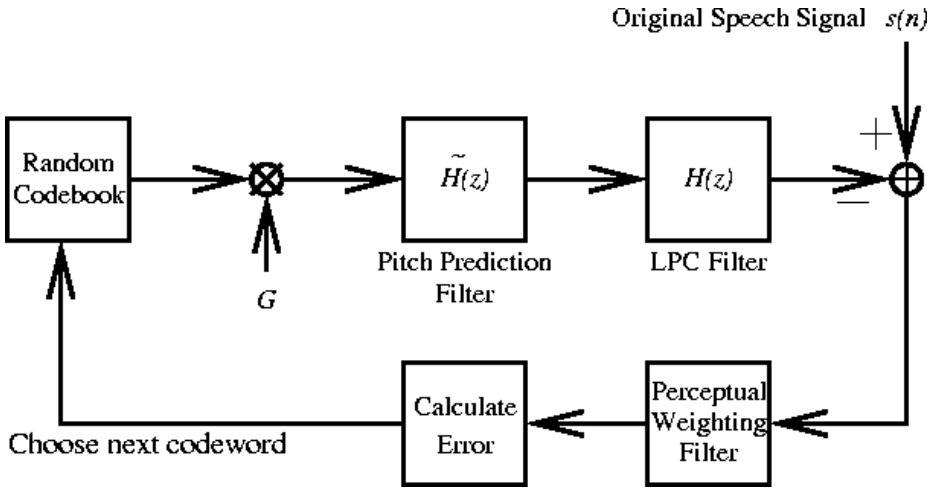


Figure 16: Block diagram of a CELP coder

6.4 Observations from previous speech coders

Waveform coders such as pulse code modulation, differential pulse code modulation, adaptive differential pulse code modulation and their variants [2, 17, 18, 33, 41] achieve high speech quality and intelligibility at high coding rates of 48 kbps or more. Speech vocoders yield smaller coding rates between 2 to 6 kbps though their perceptual quality is lower than that of waveform coding. Vector quantization can be applied to both of these types of speech coders to achieve the same kind of quality [6].

6.5 Experimental procedure of LCR on actual TIMIT speech data

Real speech corpi from the TIMIT database are used to test the LCR algorithm. The TIMIT speech database is composed of male and female speakers from different dialect regions. In our experiment, female speech samples were obtained from 25 different speakers belonging to dialect region 1. Eight speech samples from each speaker were used for training of the

LBG codebook to be later used in our LCR algorithm. This is equivalent to 200 different speech samples. Speech samples from 10 speakers from the same dialect region were used to test the LCR algorithm. The main tool used to perform the simulation was MATLAB.

Both training and testing vectors are obtained as line spectral pair coefficients. Vectors consisted of linear prediction data expressed as line spectral pairs. These were obtained by breaking the speech into frames of 96 samples from which 10 line spectral pairs were obtained. From these sets of speech corpi, an 247688 x 10 training and 107236 x 10 testing dataset of line spectral pair (LSP) coefficients were obtained.

Having acquired LSP coefficients for speech, we set out to compare rate-distortion performances of FSVQ, LCR, DCR, truncated LCR, and truncated DCR algorithms on real speech data. We truncate the re-ordered codebooks down to 1/32 of the original codebook size. We apply the following procedure to the speech dataset.

1. Codebooks are obtained from the training set of TIMIT speech and Gauss-Markov data for a given codebook size ranging from values of 32 to 1024 as multiples of 2.

2. Once the codebook is generated, the training data is used to calculate the transition matrix for both LCR and DCR algorithms.

3. For every given data vector, we find the absolute index of the previously encoded data vector.

4. Truncation takes place by accepting the most $likely$ (from LCR algorithm) or the most $similar$ (from DCR algorithm) components of the reordered codebooks respectively. (see section 4.4)

5. Rate-distortion graphs are obtained for the truncated versions of LCR and DCR and compared to FSVQ.

   - Signal-to-quantization-noise (SQNR) (in dB) = $10 \log_{10} \frac{\sigma_{X_{seq}^2}}{d}$
     where $d = \sum_{i=1}^{N} \frac{(X_{seq}(i) - Q(X_{seq})(i))^2}{N}$,

N=number of testing data vectors,

X_{seq}=sequence of input data in 1-d,

d is distortion per dimension.

$\sigma_{X^2_{seq}}$=variance of input data sequence = var(X_{seq})

$Q(X_{seq}(i))$ is the encoded version of X_{seq}

- Rate $r = \log_2 C_t$ where C_t is the size of the number of code vectors in the truncated codebook.

6. Total bit rates for the rate-distortion curves for both speech and Gauss-Markov data are obtained as follows.

   - Find the reordered indices for codebook size C_t
   - Find the bits for size C_t, $B_e(k)$= Huff (I(k)) where Huff(i) is the Huffman encoded bits assigned to i in the data reordered indices sequence
   - Total bit rate per sample for entropy coding is $\sum_{i=1}^{N} \frac{B_e(k)}{N*d}$

7. Standard bit coding rate for LCR, DCR, without Huffman coding is $B_s = \frac{\log_2 C_t}{d}$

8. Apply ECVQ, FSVQ, on TIMIT speech data and observe rate-distortion curves

9. Apply Arithmetic coding on the VQ *unreordered* indices

6.6 Speech coding results

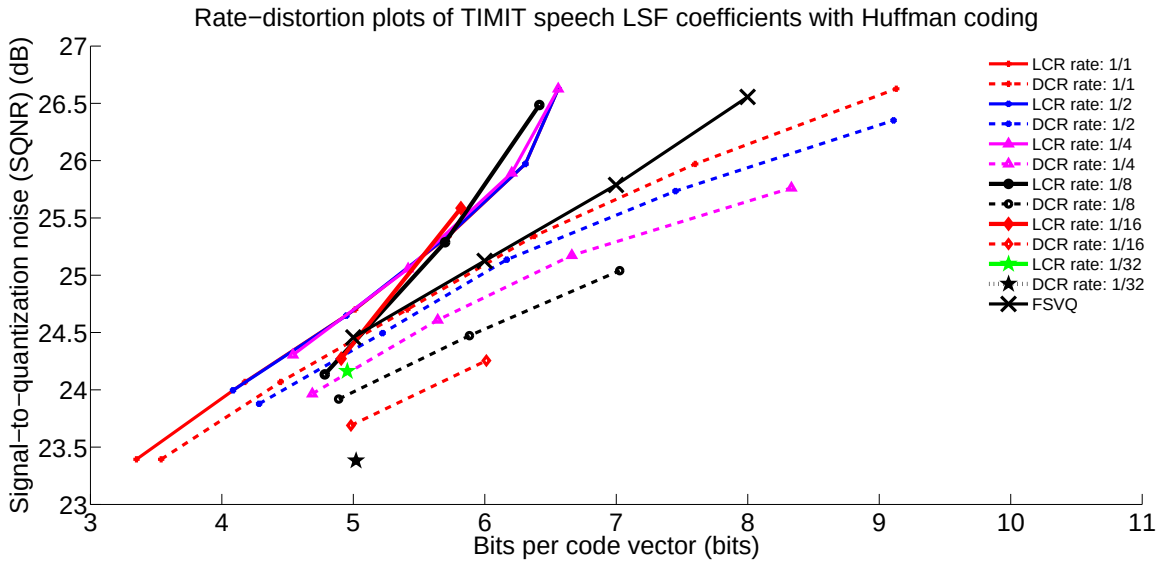
In Figures 31,17 and 19, the rate-distortion curves are obtained by observing the effect of changing the code book size on the signal-to-quantization noise (SQNR) for real speech data. All figures clearly highlight that LCR compresses better than DCR for all levels of truncation as explained in section 4.4. More evident is the fact that the LCR codebook truncated by a factor of $\frac{1}{16}$ outperforms the state-of-the-art finite state vector quantization algorithm. This practically means, we can save the computational complexity or just practical complexity of implementing an FSVQ algorithm by simply truncating an existing VQ

using the LCR scheme. This may be very useful for all forms of data especially speech data.

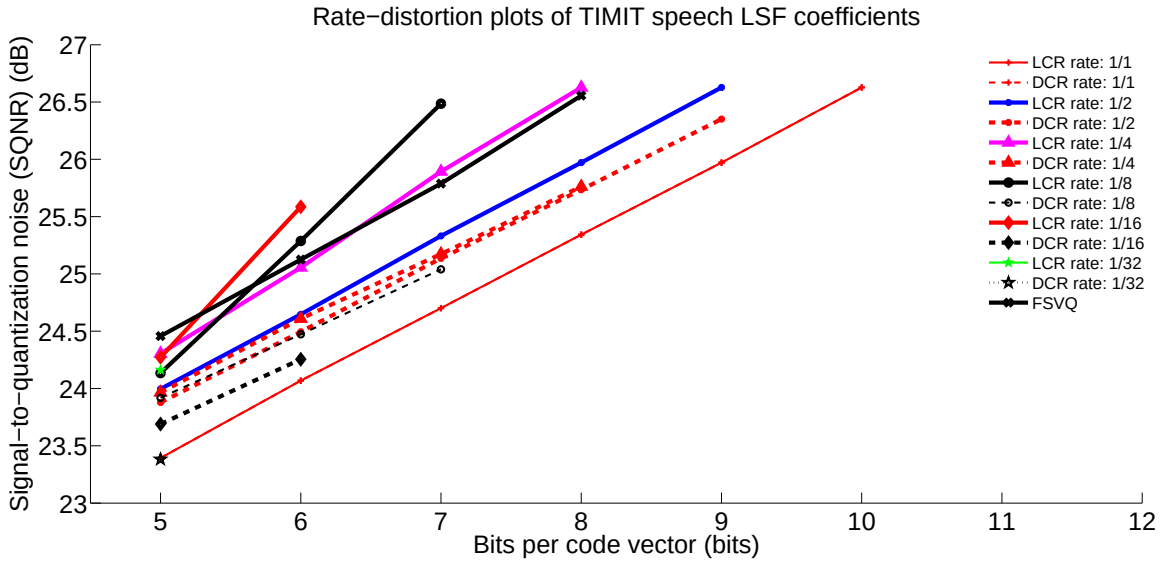
Figure 18 compares Arithmetic coding, likelihood codebook reordering VQ, and dynamic codebook reordering VQ on speech LSF coefficients. It shows that LCR outperforms DCR on re-ordered indices and Arithmetic coding on the absolute indices of the vector quantization codebook in terms of rate-distortion performance. For a given signal-to-noise ratio, our algorithm saves about 4 bits relative to Arithmetic coding on the un-reordered set of indices.

Figure 19 is a comprehensive figure that compares 5 major vector quantization schemes: vector quantization, finite-state vector quantization, likelihood codebook re-ordering vector quantization, dynamic codebook vector quantization, and entropy-constrained vector quantization. In this figure 19, we observe the effect of entropy coding on rate-distortion performance on the VQ indices as the size of the dictionary codebook is increased. Of all the VQ schemes, Figure 19 LCR has the highest SQNR compared to DCRVQ, LCRVQ, ECVQ, and FSVQ. The LCRVQ algorithm and all its truncated versions perform better than Arithmetic coding on our LCR algorithm.

Summarily, LCR is a new method of loss less compression that yields great savings in speech compression. It does so without introducing additional coding delays, or significant increase in computational costs. The LCR algorithm compresses even better when the codebook is truncated according to section 4.4 and Huffman encoded. We hope that this method can be incorporated in coding standards that utilize code-exited linear predictions, mixed-exited linear prediction and their variants.



(a)



(b)

Figure 17: a) Rate-distortion curve of Speech vs distortion with Huffman coding b) Rate-distortion curve of speech without Huffman

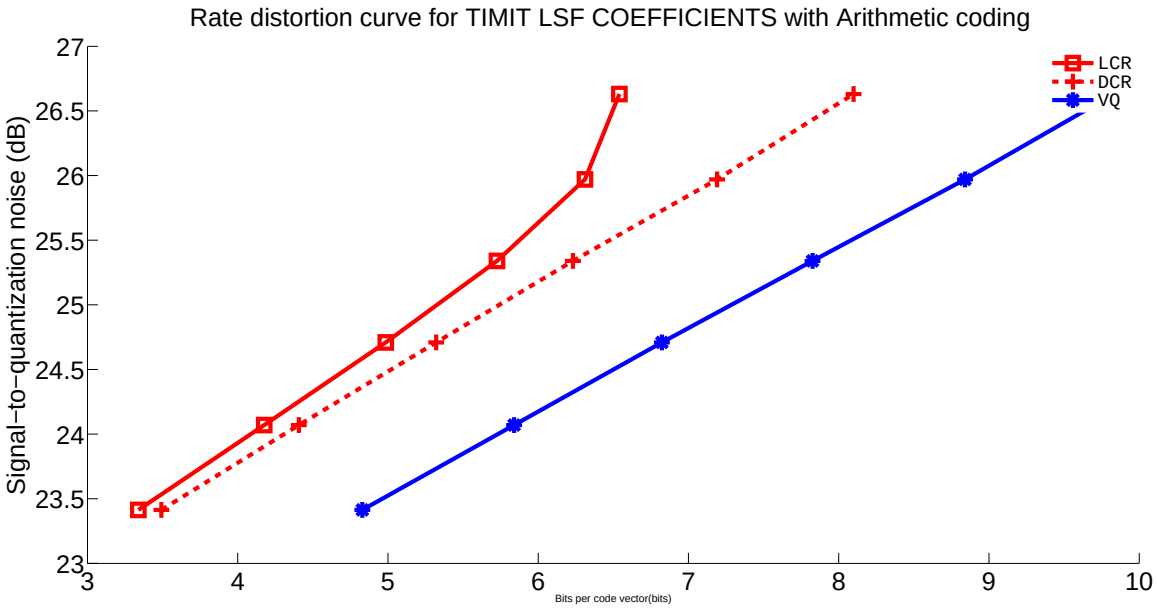


Figure 18: Rate-distortion plots of Speech LSF coefficients with Arithmetic coding

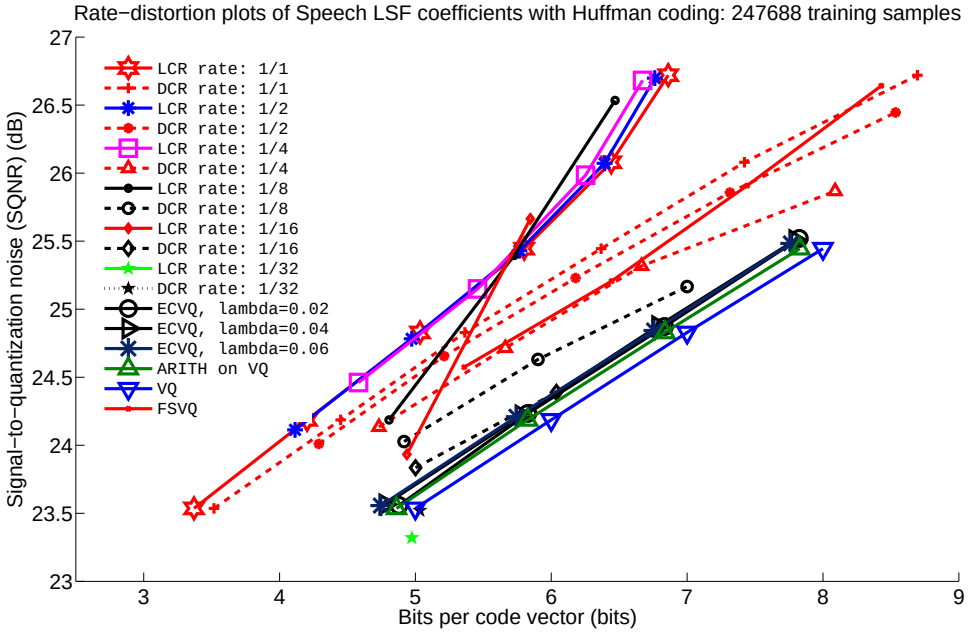


Figure 19: Rate-distortion plots of various VS schemes on Speech LSF with entropy coding

CHAPTER VII

APPLICATION OF LIKELIHOOD CODEBOOK REORDERING ON IMAGES

7.1 Image compression using vector quantization

Image compression is vital for many applications such as TV transmission, video conferencing, facsimile transmission of graphic images, transmission of images from satellites and reconnaissance air crafts. These application of image coding necessitate a need to efficiently transmit and encode images at an optimal bit rate and minimal computational complexity. A fundamental goal of data compression is to reduce the coding bit rate for transmission or data storage while maintaining an acceptable level of image quality. Many compression techniques have developed such as differential pulse code modulation, transform coding, hybrid coding, and adaptive versions of these techniques in response to the need compression images. These aforementioned methods usually exploit the psychovisual as well as statistical redundancies in the image data to reduce bit rate. One shortcoming with these methods is the quantization is performed on a sample-to-sample basis or pixels for images. Since the neighboring image pixels are typically correlated, these scalar quantization methods are not optimal. Shannon's rate distortion theory propounds that a better performance is always achievable in theory by coding vectors instead of scalars, even though the data source is memory-less.

For any vector quantization algorithm, the following three criteria are used to evaluate its performance:

- Coding rate
- Distortion, signal-to-noise ratio, or perceptual quality

- Computational complexity

When vector quantization schemes with memory are applied to speech or images, the coding rate for a given signal-to-noise ration drops by approximately half. Finite-state vector quantization and predictive vector quantization, for example, achieves a bit rate of .375 bits per pixel instead of 0.7 bits per pixel as compared to vector quantization for a reasonable level of image quality. Finite-state vector quantization, predictive vector quantization, and adaptive finite-state vector quantization all achieve low coding rates but a price of increased computational complexity. This stated fact is exacerbated especially when the vector dimensions are greater than five.

Dynamic codebook re-ordering vector quantization was introduced whereby indices of codebook vectors are re-ordered such that the majority of the new set of re-ordered indices are close to 0. This DCR scheme is a post vector quantization method as it produces the same image as a memory-less vector. DCR is advantageous because it minimizes both computational cost and coding rate simultaneously for the same level of distortion. In [10], the likelihood codebook re-ordering vector quantization approach was applied to both Gauss-Markov data and TIMIT speech data. Results of this experiment showed that LCR outperformed DCR in terms of entropy coding rate. In this chapter, we present the experimental procedure to apply likelihood codebook re-ordering for images. It is necessary to observe the effect of this scheme on images because images are two-dimensional and the concept of finding the next vector to be quantized is ambiguous since for every image block, the next vector could be one of eight vectors as opposed to the one-dimensional nature of speech signals.

7.2 Experimental procedure on Images

Images from the University of Southern California Signal and Image Processing Institute (USC-SIPI) are used to build the training vector database for image vector quantization. To obtain the transition matrix for the LCR algorithm, the transition matrices for each of the

10 images available in the training dataset are summed up. The re-ordering matrix for the image dataset is consequently obtained from the overall transition matrix. This re-ordered matrix is then used to obtain the re-ordered indices for our test image. For our experiments, the images are operated on in the luminance-chrominance plane also otherwise know as the YUV plane. Our codebook design, training and testing of LCR and DCR algorithms are applied to the luminance component (Y) of the both the training and testing images.

The image training dataset has 40960 blocks, each of size 4 by 4 while our testing dataset has 499392 blocks of the same 4 by 4 dimensions as the training dataset. Each of these blocks are converted to vectors whereby the 4 by 4 blocks are converted to 16-by-1 vectors. The procedure used to experiment on our data is as follows:

1. Codebooks are obtained from the training set of the USC-SIPI images by using the Luizo-Buzo-Gray codebook design algorithm.

   - Given X_{train}=$\{X_1, X_2, X_3, \ldots, X_T\}$ where X_{train} is training dataset, X_i is the ith vector of the training set, and $T$is the size of the training dataset.
   - Also assume x_{test}=$\{x_1, x_2, x_3, \ldots, x_L\}$ where L is the size of the testing dataset, x_{test} is the testing dataset, and x_{test} is the testing dataset.
   - The codebook $C = \{c_1, c_2, c_3, \ldots, c_K\}$ is the set of vectors that minimize the following condition:

$$D_{ave}=\frac{\sum_{i=1}^{T} \|X_i - Q(X_i)\|}{MK}$$

$$\text{where } c_k = Q(X_i),$$

   D_{ave} is the average distortion.

2. Once the codebook is obtained, the training data is used to calculate the transition matrix for the re-ordering algorithm.

3. The absolute indices of the testing image blocks are obtained by finding the nearest neighbor codebook vector closest to the testing image block. This is done as follows:

The distance between x_i and c_k , $\|x_i - c_k\|$, is calculated for all k code vectors, and the index of k which corresponds to the smallest distance is assigned to the absolute index of a given test image block.

4. The re-ordered matrix is obtained by sorting the row of each transition matrix such that the most likely codebook vectors are assigned index of 0. Subsequent code vectors with smaller likelihoods are assigned indices of increasing order. Truncation can be performed on this re-ordered codebook for both LCR and DCR algorithms.

5. Rate-distortion curves are obtained for the truncated versions of LCR, DCR, FSVQ, and VQ algorithms.

6. Entropy-distortion plots are also obtained by calculating the sample entropy of the coder at a given codebook size.

7.3 Results

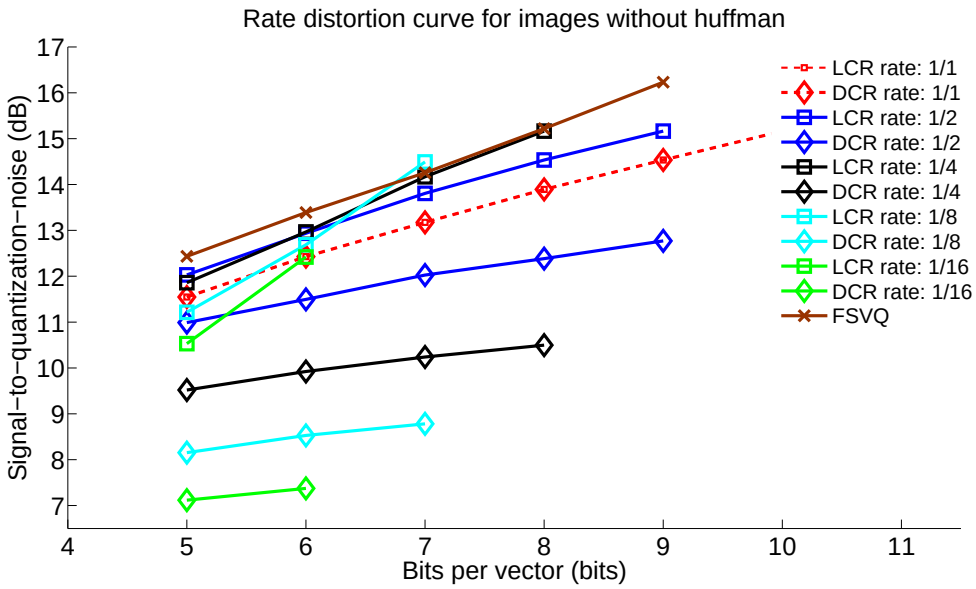


Figure 20: Rate-Distortion curve OF IMAGE VQ indices without Huffman

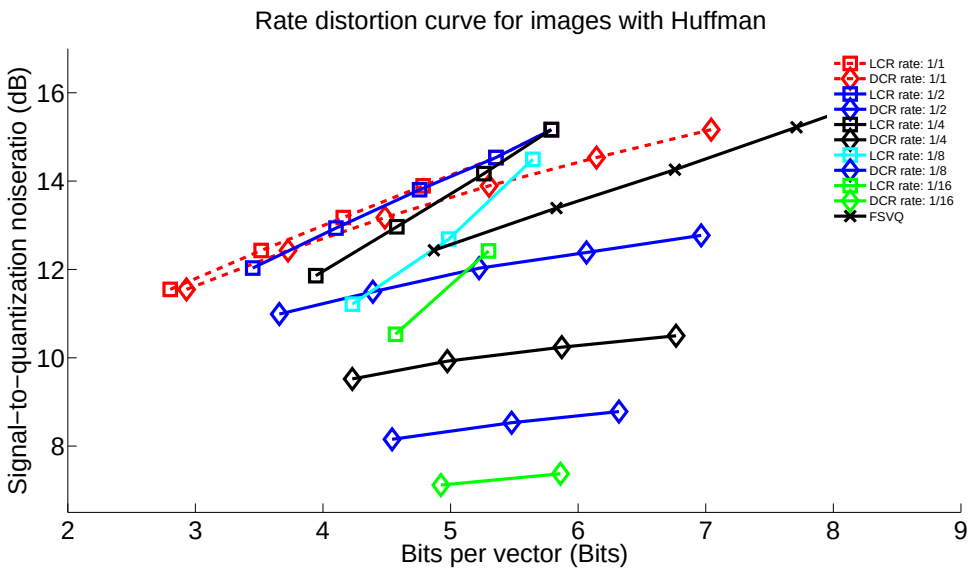


Figure 21: Rate-distortion curve of image indices with Huffman

7.3.1 Interpretation of results

Section 7.2 explains the experimental procedure to obtain the rate-distortion plots for the images. Figure 20 shows the relative performance of LCR vs DCR appended by Huffman coding for different codebook sizes. It is observed that LCR clearly outperforms DCR for all levels of truncation. The truncation is performed using the approach outlined in section 4.4. Figure 20 shows that distortion increases as the truncation is increased for both LCR and DCR. The distortion caused by DCR truncation increases much more than that of DCR because the probability distribution of LCR for the re-ordered indices approaches zero much faster than that of DCR. This simply means that more of the re-ordered indices are closer to zero for LCR than DCR indices.

Without Huffman coding, truncating the codebook of an LCR scheme yields a better rate-distortion performance while reducing the rate-distortion performance of its DCR equivalent. This effect is shown in figure 21. In essence, the LCR truncated algorithm achieves a smaller distortion and hence larger signal-to-noise ratio than DCR algorithm for the same coding rate when the re-orderings are not appended by Huffman coding. It is

evident that LCR benefits more from truncation of images than the DCR algorithm.

Figures 22, 23, 24, 26 are figures that visually demonstrate the power of the LCR algorithm. It can be seen that LCRVQ with truncation by 1/8 as shown in Figure 26 yields almost the same picture as that of the originally VQ quantized image. This shows that our algorithm in addition to reducing the compression rate preserves the perceptual quality of the image signal for truncation up to 1/8. Truncating LCR by factors less than 1/8 yields more visible degradation. Therefore, one can observe a trade-off between degradation in perceptual quality and the coding rate.

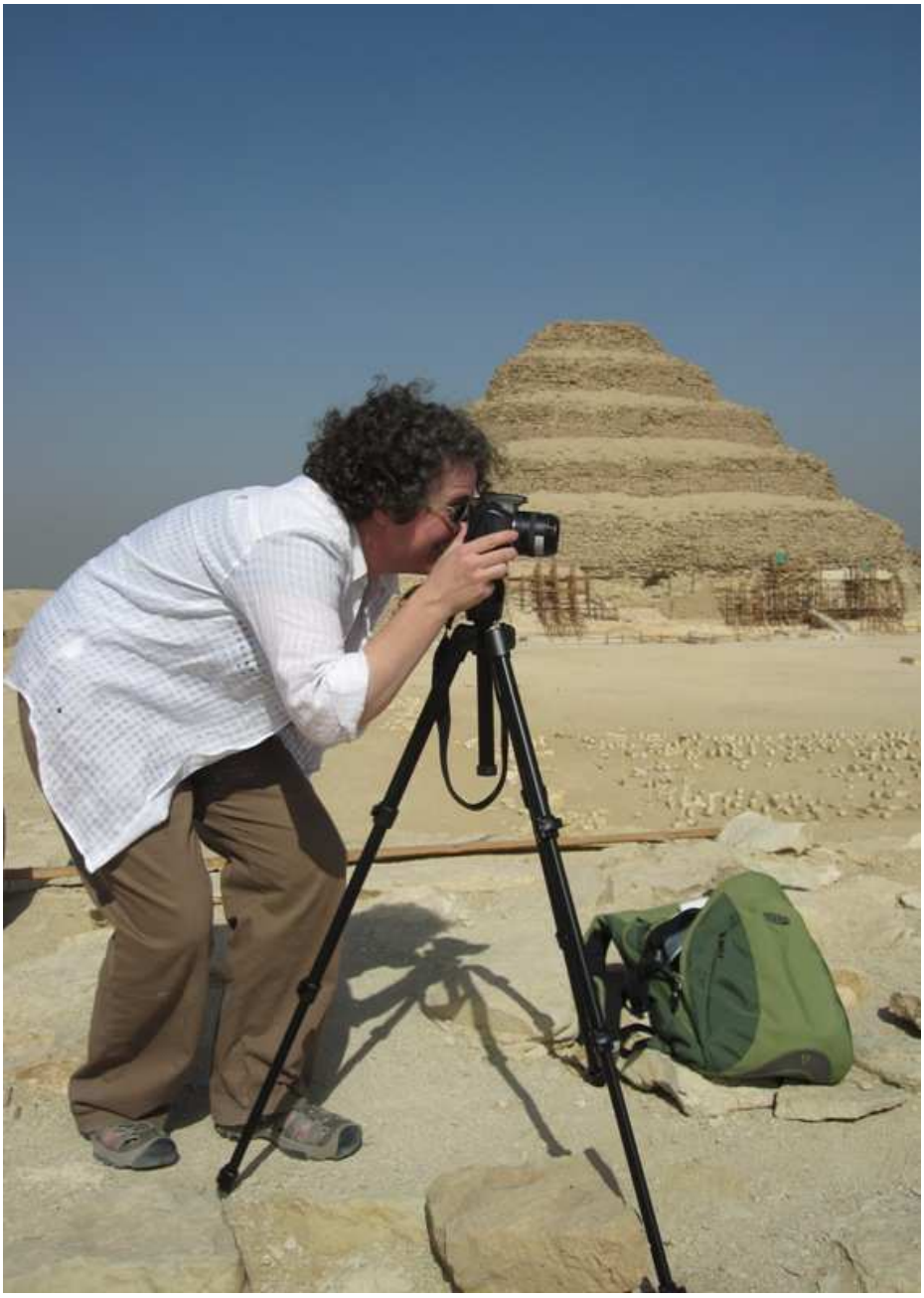


Figure 22: ORIGINAL IMAGE WITHOUT COMPRESSION

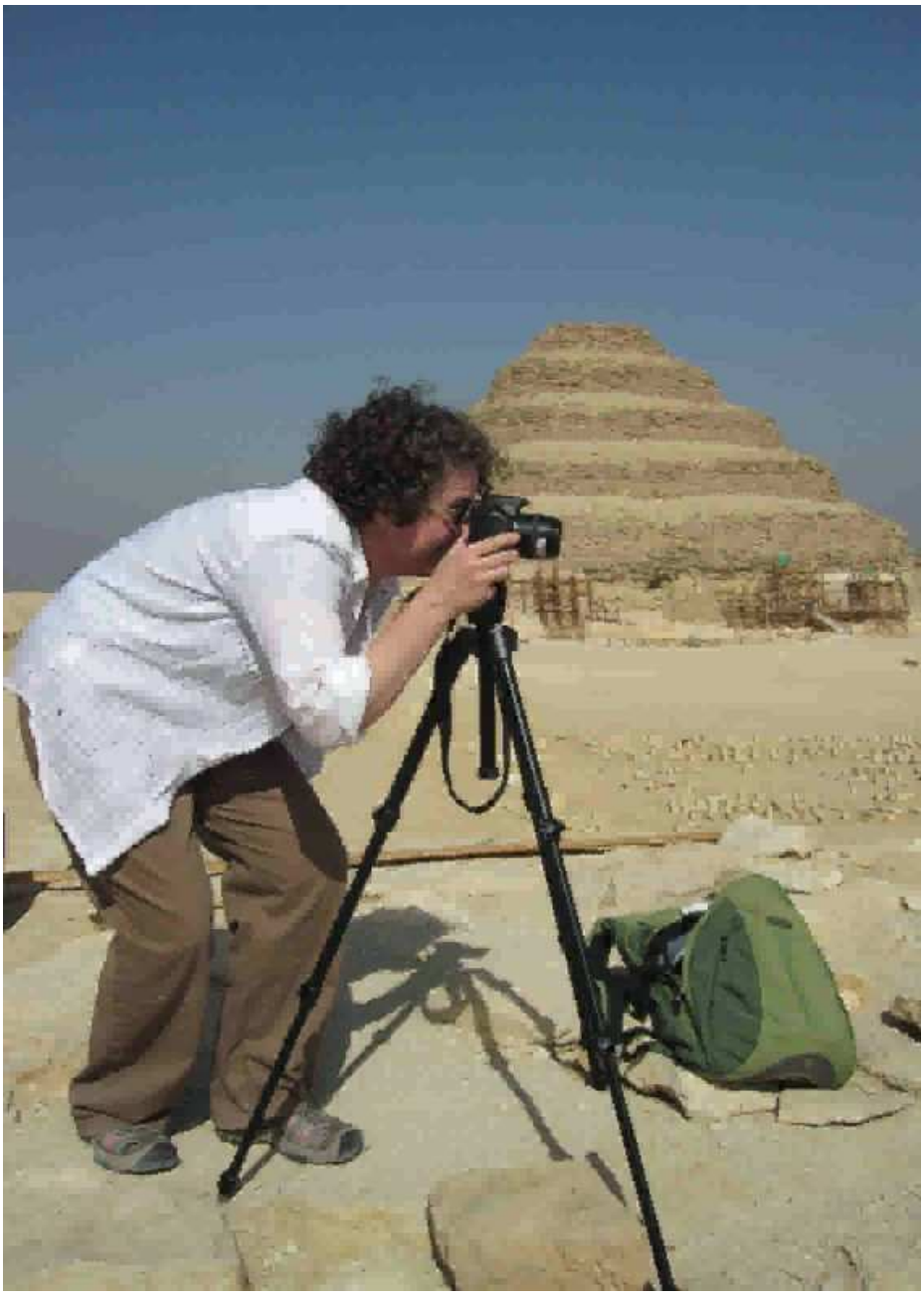


Figure 23: ORIGINAL IMAGE WITH VQ COMPRESSION (0.625bpp)

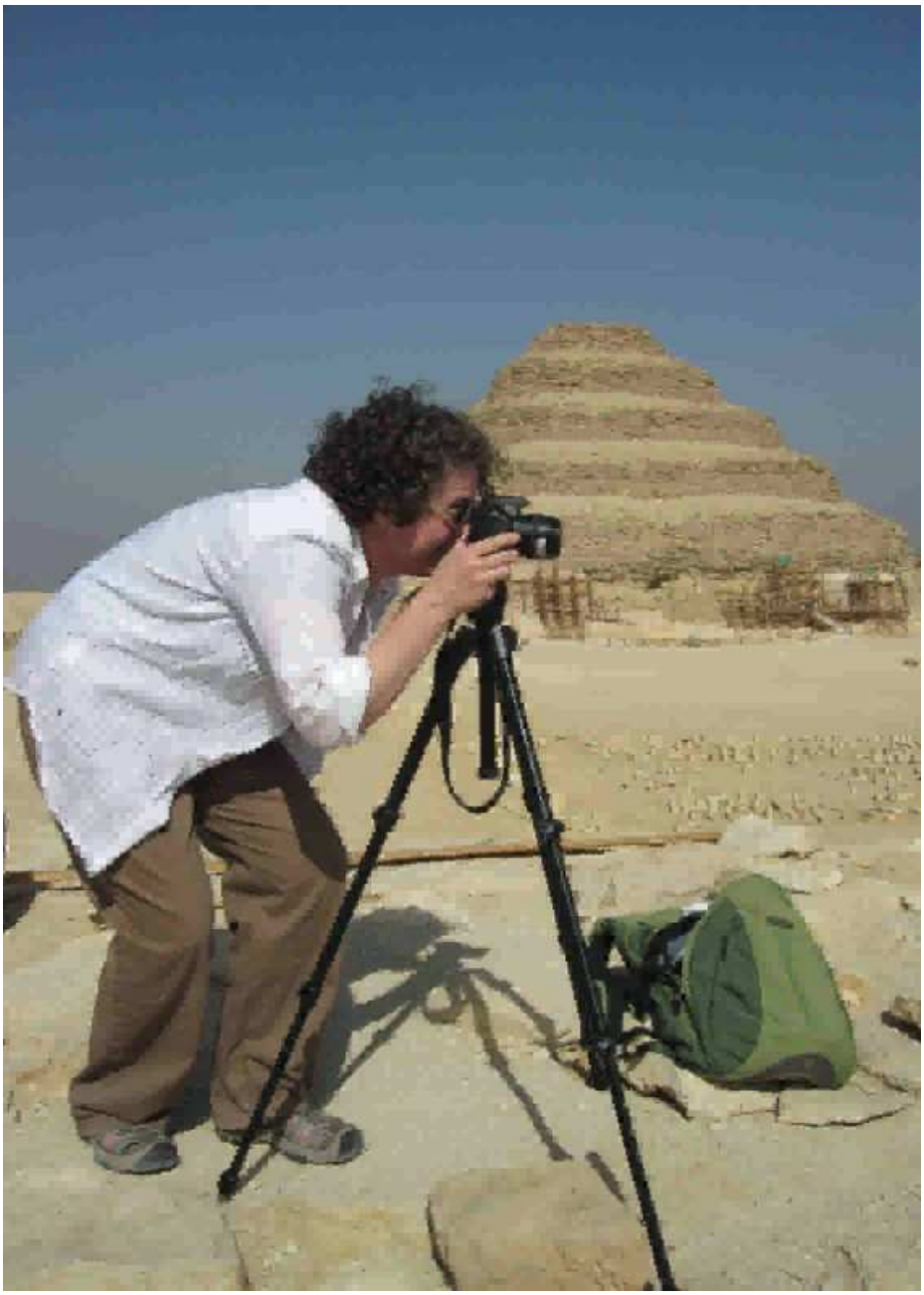


Figure 24: ORIGINAL IMAGE WITH LCR VQ 1/4 COMPRESSION (0.5bpp)

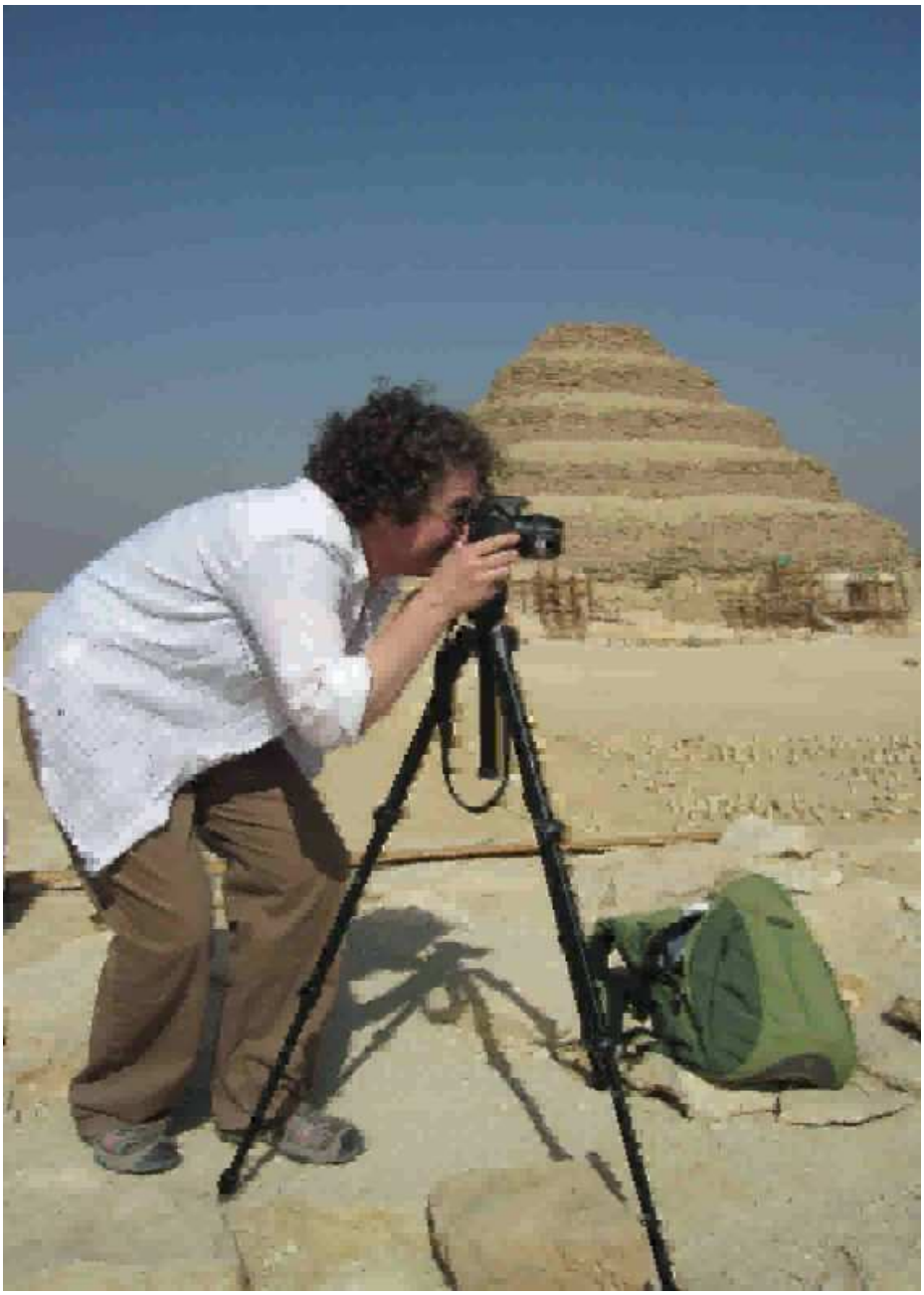


Figure 25: ORIGINAL IMAGE WITH LCR VQ 1/8 COMPRESSION (0.4375bpp)

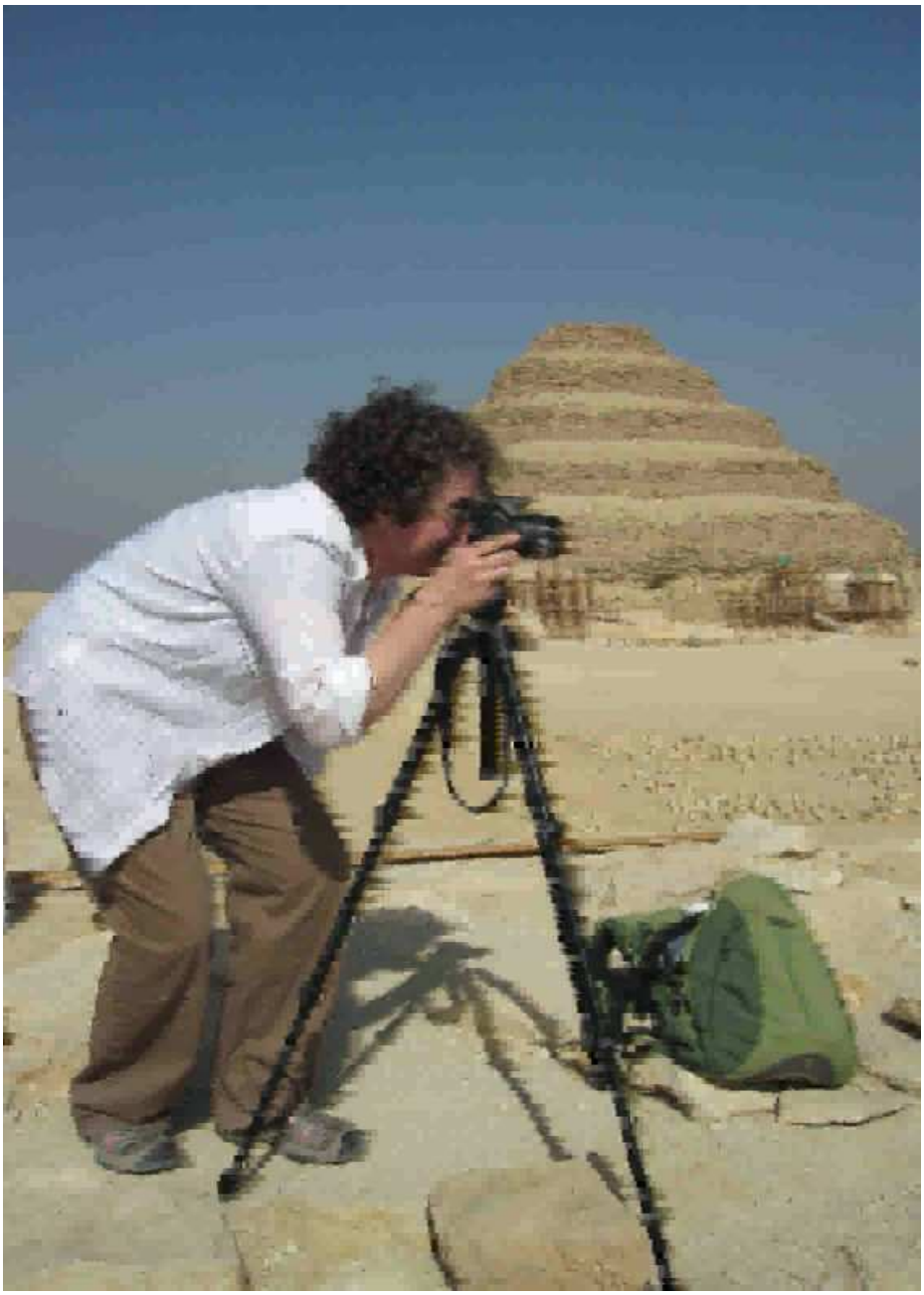


Figure 26: ORIGINAL IMAGE WITH LCR VQ 1/16 COMPRESSION(0.375bpp)

CHAPTER VIII

ONLINE LEARNING FOR LIKELIHOOD CODEBOOK REORDERING VECTOR QUANTIZATION

In order to encode data using likelihood codebook reordering vector quantization, we need to know the transition matrix of the data to be encoded. However, the transition matrix for the data to be compressed is not available *a priori.* The transition matrix for encoding a test data is different from the transition matrix obtained from an initial training data set. This mismatch in the transition matrices requires one to either accurately estimate the true transition matrix of a given test vector sequence assuming knowledge of the transition matrix from the training vector sequence. The process of estimating the true transition matrix on an online basis is called *adaptive likelihood codebook reordering vector quantization* [30]. To accurately encode the test vector sequence, it is imperative to know the transition matrix of the data vector sequence to be encoded. The challenge in most real encoding systems is that data vector sequences are encoded one vector at a time. It is hence not possible to have a real estimate of the transition matrix of the data to be encoded ahead of time.

The transition matrix T of the testing data sequence is initialized in one of two ways

1. The transition matrix is initialized by assigning all values of the transition matrix to a single value. 1 is the default value though other positive numbers will perform the same role. In a nutshell, the following is performed:

$$T(i, j) = a \tag{8}$$

   for all i, and j where $a > 0$

2. The transition matrix is initialized to the transition matrix obtained from the training

data set. $T = T_{train}$

where T_{train} is the transition matrix of the training data set. After the transition matrix is initialized, the transition matrix is estimated for every encoded data vector as described in Section 8.2.

8.1 Entropy estimates

In the ALCR-VQ compression scheme, each data vector is encoded by using the transition matrix at a given time instant. Typically, entropy is calculated using the following equation:

$e = \sum_i p(i) \log_2 p(i)$

where $p(i)$ is the probability of occurrence of the code vector with index i. This definition of entropy suffices when the probability distribution of the code vectors is known ahead of time. In this chapter, we refer to this definition of entropy as sample entropy because it is based on the actual count of the vectors that are trained and tested. In our ALCR-VQ algorithm, the probability distribution of both the unordered and ordered indices are not know a priori. It is important to have a sense of how well the ALCR-VQ is encoding the data vectors and how close the transition matrix is approaching the real transition matrix of a data set. To define the entropy for our adaptive likelihood codebook reordering vector quantization system, the concepts of short-term and long-term transition matrices are introduced. The short-term transition matrix is the transition matrix estimated at a given temporal or spatial instant of the test vector sequence. The short-term transition matrix is obtained in the following way:

1. Short-term transition matrix $T_{short}(i, j, 0)$= T_{train} or $T_{short}(i, j, 0) = a$ for $a > 0$.

2. For t=1 to N

   $T_{short}(i, j, t) = T_{short}(i, j, t) + \delta$ where δ is the fixed increment in the transition matrix element

The long-term transition matrix (LTTM) T_{inf} is obtained by assuming we can calculate the transition matrix of the whole data vector sequence ahead of time. Hence, the LTTM is obtained by performing the standard transition matrix evaluation for sequence.

When performing the ALCR-VQ algorithm, the short-term transition matrix gives us an idea of how many transitions between code vectors have occurred for the past encoded data vectors. We can get a sense of how well our adaptive approach is performing by entropy coding the transitions between the code vectors. Our entropy evaluation is given below:

$$e(t) = -\sum_{j=1}^{N}\sum_{i=1}^{N} T_{inf}(i,j)\log_2 T_{short}(i,j,t), \tag{9}$$

$$T_{inf}(i,j) = \frac{m^f(i,j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} m^f(i,j)} \tag{10}$$

$$T_{short}(i,j,t) = \frac{m(i,j)}{\sum_{i=1}^{N} m(i,j)} \tag{11}$$

where $m(i,j)$ and $m^f(i,j)$ are counts of the transitions from dictionary vector c_i, to vector c_j in the short-term and long-term transition matrices. Equation 9 is calculating the entropy, the only difference being that the we use the long-term transitions as the real frequency of transition between two code vectors instead of the short-term transitions. The short-term transition matrix simply serve to give one an estimate of how many bits will be needed to encode the transitions at given time instant. This is evident in Equation 9 as it has both long-term and short-term matrices. Subsequent results will reveal the power of this tool and logic in quantifying compression performance. When Equation 9 is used calculated assuming that the short-term matrix is equal to the long-term matrix, we dub the entropy estimate as *predicted entropy*. This term is coined as such because it based on solely on the transition probabilities of the code vectors. The entropy estimate is not based on the probability of occurrence of indices.

8.2 Adaptive Likelihood Codebook Re-ordering Vector Quantization

The algorithm of likelihood codebook reordering vector quantization (LCR-VQ) is described in [10]. The adaptive likelihood codebook reordering vector quantization (ALCR-VQ) addresses the mismatch in the transition matrices obtained from training and testing datasets [30]. The testing datasets is typically streamed one vector at a time. There is therefore no *a priori* knowledge of the transition matrix from v_{t-1} to v_t. Therefore at every given time, t, the transition matrix and its corresponding reordering matrix are updated. The online estimation of the transition matrix of the testing data sets have the advantage of reducing the coding rate per data vector relative to when there is no adaptation of transition matrix. The modern computational resources and the speed of the modern systems overcompensate for the computational demand of the online estimation of the transition and reordering matrices. The algorithm for performing ALCR-VQ is outlined in [30] and below.

Consider the sequence of vectors $X[t]$ exhibited as $x[t]$. Each input vector sequence $x[t]$ is most closely approximated by some codebook entry $c_i[t]$. Thus a data sequence $\vec{X}$ is encoded into a sequence $C[t] = \{c[0], c[1], c[3], \ldots, c[K]\}$ where $c_i \in \mathcal{C}$.

The procedure used for generating the likelihood codebook reordering framework is summarized as follows:

1. Using a set of training vectors, create a codebook following established methods such as k-means clustering.

2. For vectors $X[t-1]$ and $X[t]$ in the training set, find the closest corresponding code elements $C[t-1] = c_{i_{t-1}}$ and $C[t] = c_{i_t}$, and note the corresponding indices.

3. Create an adaptive transition matrix $\mathbf{M}_\Delta$ and initialize to a random matrix. Sometimes, the adaptive transition matrix is initialized to the training transition matrix.

The reordered index is computed as follows for every the encoded version of vector $X[t]$ at time t.

1. Find the codebook entry c_{i_t} that most closely resembles $X[t]$ where $Q(X[t]) = c_i$ and the codebook entry $c_{i_{t-1}}$ that matches the vector $X[t-1]$.

2. Increment the appropriate transition entry (i_{t-1}, i_t) in $\mathbf{M}_\Delta$ by incrementing by some Δ according to some policy, heuristic or analytical. Δ is assigned 1 for our experiments for simplicity. $\mathbf{M}_\Delta(Q(X[t-1]), Q(X[t])) = \mathbf{M}_\Delta(Q(X[t-1]), Q(X[t])) + \Delta$

3. Sort the $(i-1)_{th}$ row of the transition matrix $\mathbf{M}_\Delta$ in decreasing order and find the order of $\mathbf{M}_\Delta(i_{t-1}, i_t)$ in the sorted row.

4. The reordered index $I_r[t]$ is given by $O(\mathbf{M}_\Delta(i_{t-1}, i_t))$

8.3 Results

The adaptive likelihood codebook reordering vector quantization was first proposed in [30]. The VQ appach was applied to both Gauss-Markov data and speech samples from the TIMIT database[44]. The results of this procedure will applied by assuming that the transition matrix is initialized to a uniform transition matrix of 1's. The Gauss-Markov data used in our experiments is generated according to the equation below:

$$Y_t = \alpha Y_{t-1} + (1-\alpha)X_t \tag{12}$$

In our experiments, we assumed the preceding generational model of our Gauss-Markov data as in Equation 5 where $\{X_t\}$ are independent, identically-distributed Gaussian random variables, $\alpha \in (-1, 1)$ is the correlation parameter, and Y_t is the source sample at time t. Training and testing vectors are obtained by generating 10 million one-dimensional samples (10M) ($10\text{M} \times 1$) which are grouped into ten-dimensinal vectors ($d = 10$). For testing data rates, a transition matrix is generated using a large number of training samples assuming that using a sufficiently large dataset should give a transition matrix which is a close approximation of the ideal transition matrix. This dataset was composed of $80\text{M} \times 1$ training samples resulting in an $8\text{M} \times 10$ test vectors. This corpus is distinct from the one

Table 6: Table of bit rates for Gauss-Markov data. Best performing algorithms for each codebook size are highlighted. S.E means sample entropy and P.E means predicted entropy

Cbk size	VQ(S.E)	LCR(S.E)	LCR(P.E)	ALCR(S.E)	DCR(S.E)
2^5	4.74	4.52	4.50	**3.9680**	4.5322
2^6	5.76	5.477	5.44	**4.8416**	5.5379
2^7	6.76	6.414	6.38	**5.7318**	6.5471
2^8	7.83	7.38	7.34	**6.7310**	7.6206
2^9	8.90	8.35	8.30	**7.9072**	8.7457
2^{10}	9.97	9.41	9.40	**9.3775**	9.8890

used for training the encoder and is used to test the bit rate as the transition matrix adapts. In order to remove the encoding algorithm as a variable, the data rate is calculated by finding the average entropy using bit counts from the training matrix and using probabilities from the more statistically-representative testing matrix.

For the Gauss-Markov samples used, the codebook size is varied in powers of 2 corresponding to the number of bits to address the codebook. As the codebook size increases, the encoding error decreases, but more bits are required for each vector. The aim is to improve upon standard VQ by reducing the bit rate across all codebook sizes. The coding rate for a given codebook size for LCR-VQ, ALCR-VQ, and DCR-VQ is calculated by entropy coding the reordered indices of the reproduction vectors of the testing data using arithmetic coding, while the coding rate of standard VQ is calculated by performing arithmetic coding on the *absolute* indices of reproduction vector.

Figure 28 shows that for all codebook sizes ranging from 32 through 1024, the ALCR-VQ method outperforms the LCR-VQ, DCR-VQ, and standard VQ compression approaches. Since test data can vary statistically from the training set, ALCR is always able to slightly outperform LCR as it is able to adapt to local, as well as global statistics. The same Figure 28 also shows that ALCR-VQ performs at its optimal performance at a codebook size of 256.

Figures 31, 27, 28, 30 are generated by estimating the entropy from the long-term and short-term transition matrices. These aforementioned figures are obtained from a 256-entry

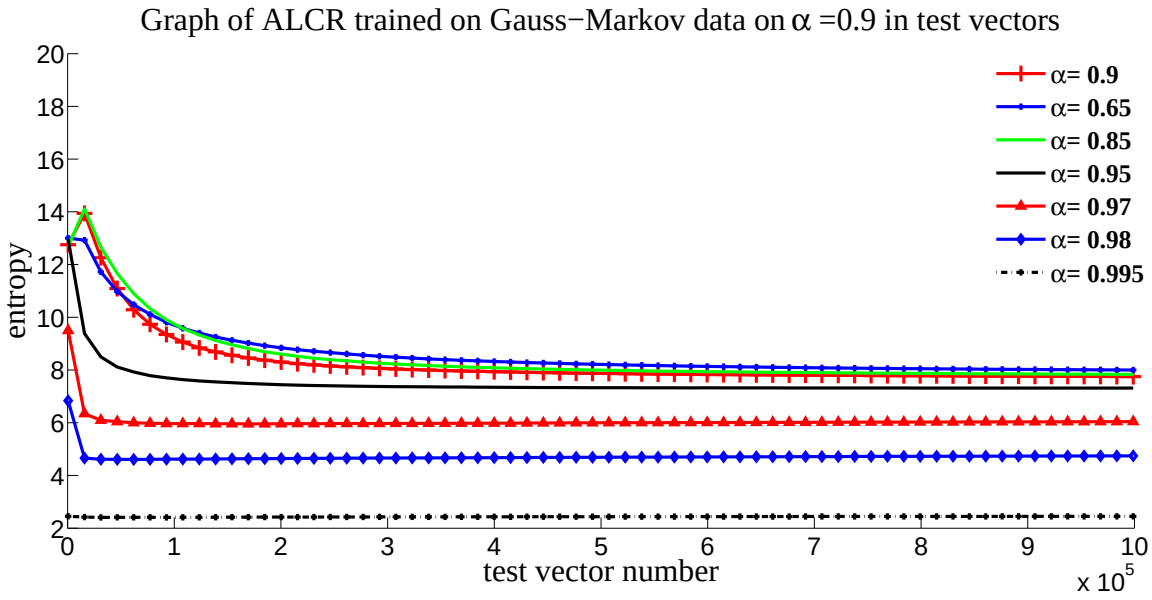


Figure 27: Testing of ALCR algorithm on Gauss-Markov source of correlation factor 0.9 on an online basis. The entropy is calculated on a 256-entry codebook and tested on 1-d Gauss-Markov data sources of correlation factors greater than and smaller than the training correlation factor of 0.9. The transition matrix is initialized to all 1's

codebook from a training data set of the correlation factor of $\alpha_{train} = 0.9$. For every input test vector, the short-term transition matrix is obtained by encoding the testing data set with the codebook from the training dataset. The entropy of the encoding system at a given instant is estimated from both the long-term transition matrix, M^f, and the short-term transition matrix, M. The entropy at vector time, t, is calculated instantaneously in Equations 9, 10, 11.

We can observe from Figures 27 the approximately pronounced exponential decay of the entropy plot for testing correlations lesser than 0.9. This is in sharp contrast to the relatively flat entropy graphs for larger entropies. The aforementioned behavior is noticed because if the testing correlation is smaller than the training correlation, there will be more code vectors that exists outside the range of the testing input vector space. The peripheral distribution of the code vectors obtained from training input vectors whose correlation is greater than the testing correlation indicates that most of the testing vectors will be mapped to code vectors within the input vector domain space. Consequently, the code vectors that

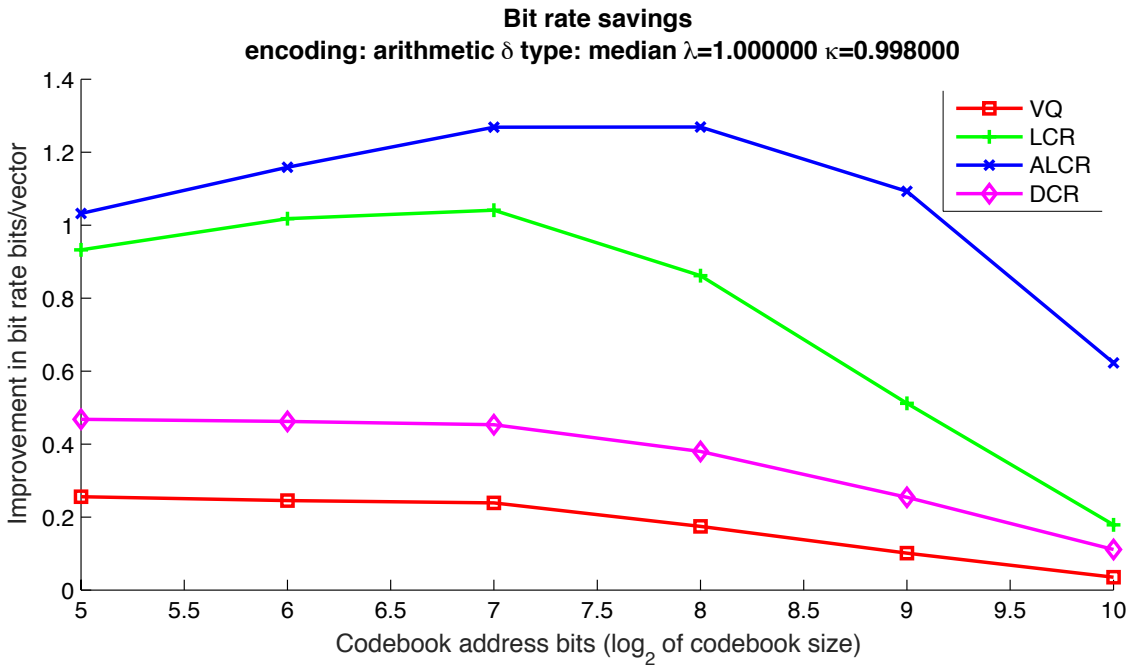


Figure 28: Testing of ALCR algorithm on Gauss-Markov source of correlation factor 0.9. We observe the improve in bit rate savings for every codebook size for each ALCR-VQ, DCR-VQ,LCR-VQ, and VQ schemes. Higher values on the y-axis infer more compression.

are outside the input domain will rarely be mapped into. This sparsity of mapping yields many entries of the transition matrix that are closer to 0 than not, which consequently implies an increased entropy for the entries with low count. The exponential decay can be explained by the fact that as more test vectors are encountered, there is an increased likelihood of testing vectors being mapped into code vectors obtained from the training data set of larger correlation. As a result, as more vectors are encountered, the cell entries of the transition matrix that were once sparse become less sparse and are consequently assigned less bits as observed from information theory.

For testing data sets whose correlations greater than the training correlation, we observe that the entropy curves takes a far less time before it saturates to a steady value. This is so because training data sets with smaller correlation than testing data set have their code book concentrated on a much smaller range of the testing input vector space. This smaller range of the code book implies that the code vectors are distributed across a smaller periphery

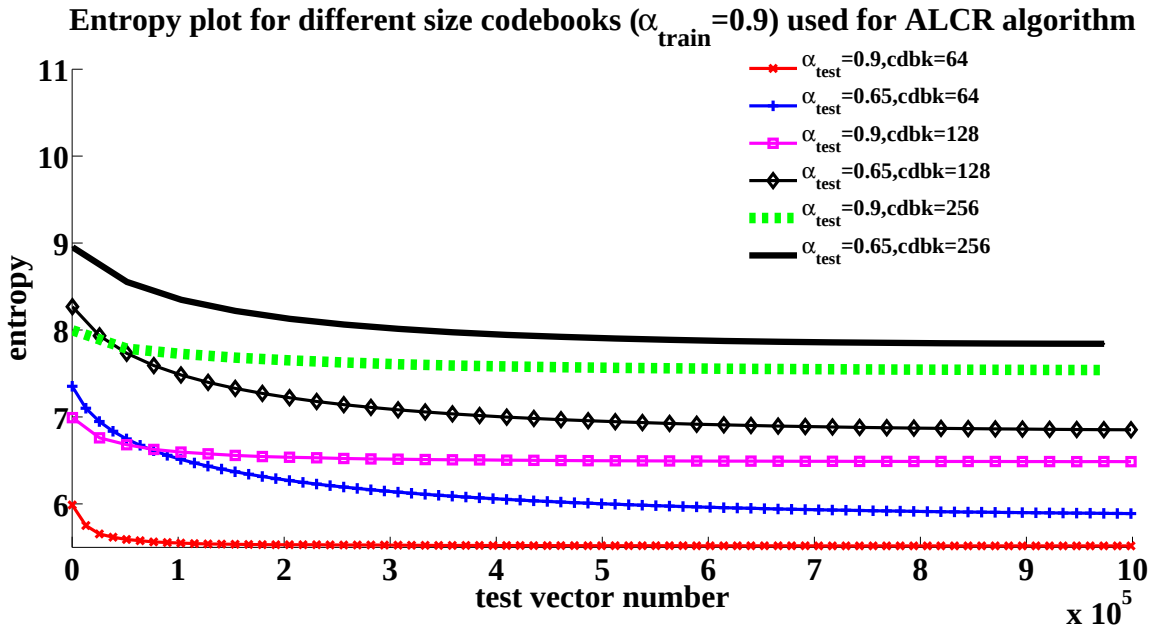


Figure 29: Evaluation of the effect of different codebook sizes on the entropy of ALCR on an online basis. Codebook sizes are 64, 128, and 256 and the testing correlation factors of the 1-D Gauss-Markov sources are 0.65 and 0.9

compared to when the correlation was larger than training vector correlation. The result of this phenomenon is the fact that code vectors are more quickly mapped to. In short, for this scenario, there are more *non-zero* transitions than zero transitions.

By observing the curves in Figure 31, it can be seen that there is an exponential-decay type of convergence approaching an asymptotic limit. Also important to note is that the entropy rate initially increases to nearly 16 bits per vector, which is nearly double the 8 bits required using a naive encoding method. This is because a small number of vector transitions have been encountered causing them to have very low bit coding but all un-encountered transitions require the new maximum entropy code size. As the transition matrix becomes increasingly representative (requiring approximately 5×10^5 vectors), the bit rate continues to drop towards its asymptote.

Figure 29 also shows that for a matched data set with correlation α_{train} of 0.9, the computed entropy approaches its asymptotic entropy bound much more quickly. This is expected since the vectors are mapped exactly into the domain of the training data set.

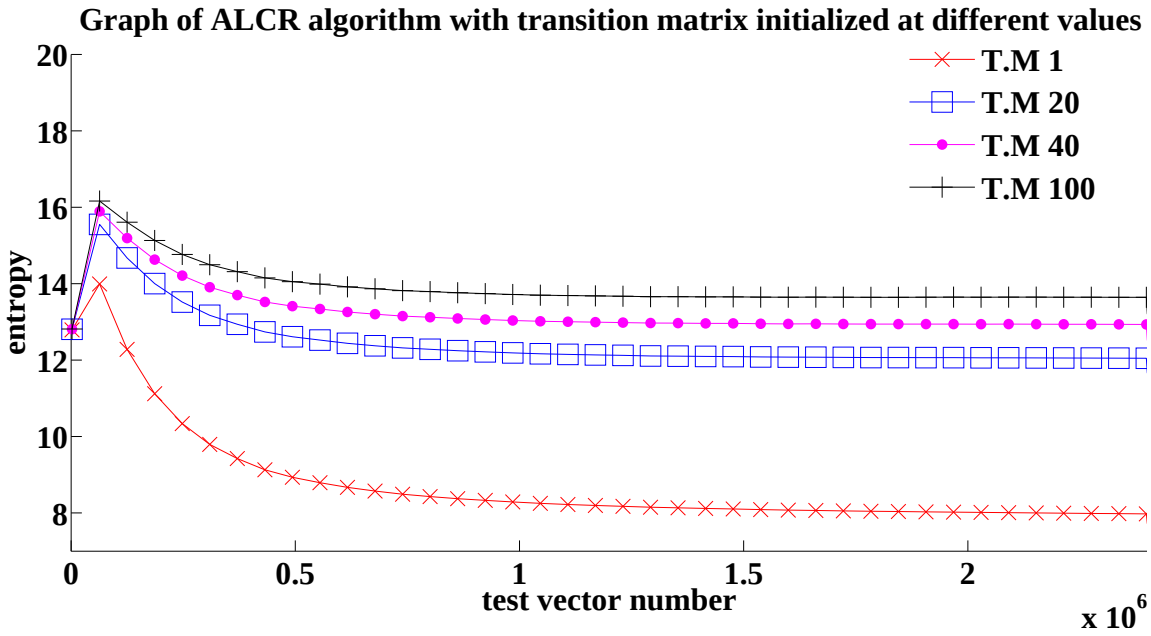


Figure 30: Testing of ALCR algorithm on Gauss-Markov source of correlation factor 0.9. We change the initialize values of the transition matrix cells to equal the numbers assigned on the legend where "T.M" stands for transition matrix initial cell values. For example, for "T.M" equal to 40, all the transition matrix values are assigned the value of 40

For all codebooks of size 64, 128, 256, and a testing factor of 0.65, there exist more code vectors that are outside the space of the input vectors. Consequently, many code vectors either have 0 mappings from the input vector space to the code vector or have significantly smaller number of mappings to the code vectors. The code vectors that are not mapped by the source vectors are not mapped frequently. Subsequently, the entropy of the system is increased.

In Figure 30, we observe the effect of initializing the transition matrix on the entropy of the ALCR-VQ algorithm. A 256-vector codebook is used to encode the testing set of the same correlation 0.9. The initial transition matrix M_Δ is used to encode the testing set of the same correlation 0.9. All elements of the initial transition matrix are set to values 1, 20, 40, and 100 as shown in Equation 13. For all values of i and j,

$$M_\Delta(i,j) = 1, 20, 40, 100 \tag{13}$$

Initializing the transition matrix at 1, the entropy takes a lot longer time to approach its

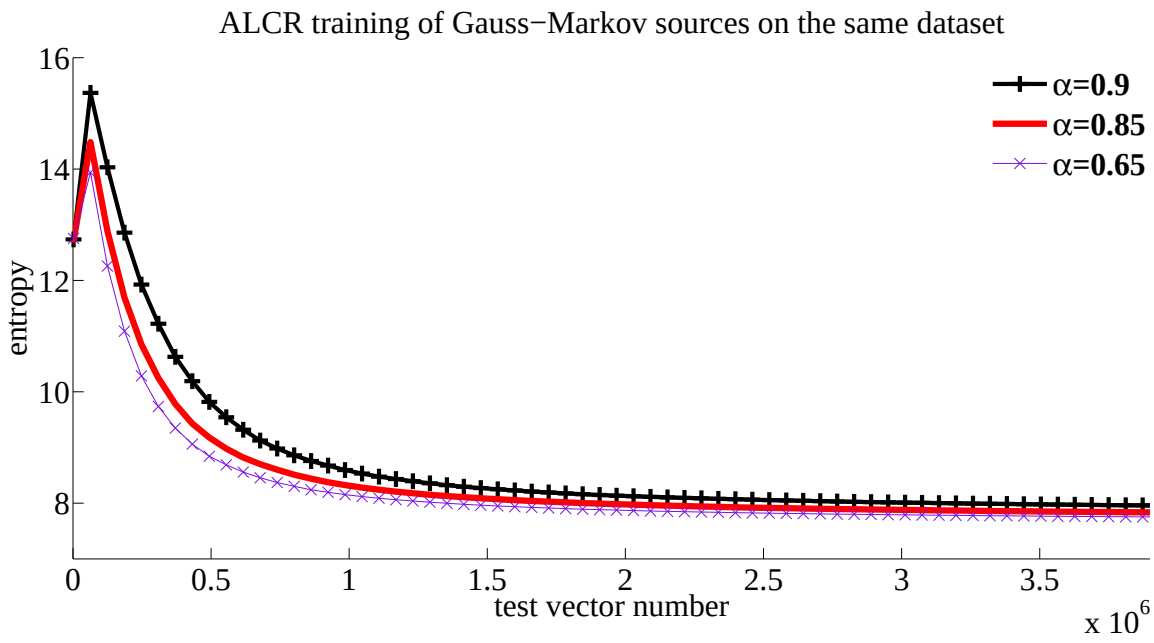


Figure 31: Cross-training and testing of ALCR algorithm on Gauss-Markov sources with the same correlation factor α

Table 7: Table of Vector quantization errors for cross-training and testing of Gauss-Markov sets

		α_{test}					
		0.65	0.7	0.75	0.85	0.88	0.9
α_{train}	0.65	**0.040**	0.041	0.044	0.058	0.071	0.090
	0.7	0.031	**0.030**	0.031	0.040	0.050	0.066
	0.75	0.026	0.022	**0.021**	0.026	0.033	0.046
	0.85	0.022	0.016	0.012	**0.008**	0.009	0.015
	0.88	0.023	0.016	0.011	0.005	**0.004**	0.005
	0.9	0.023	0.015	0.010	0.004	0.002	**0.001**

true entropy values as test values are trained on the ALCR algorithm. When the transition matrix is initialized at larger values, the entropy plot approaches higher entropy values more quickly because as the test vectors are encoded, the addition to the transition matrix still leaves it fairly uniformly distributed.

Table 7 is obtained from codebooks of datasets of one correlation factor and applied on a 1-D Gauss-Markov sequence of another correlation. We observe the effect of cross-testing on the squared rates of the testing 1-D Gauss-Markov data sequence. It is observed that the squared-error is least when the training and testing correlations match. The errors

increase the more the mismatch in the testing and training factors.

8.4 Conclusion

Because LCR Is based upon statistics rather than distance metrics, it has the capability of improving upon existing DCR-based VQ techniques. Furthermore, allowing adaptation of the transition matrix allows an even greater improvement. It can be observed from the experiments that performing ALCR-VQ on test data vectors with correlation greater than of the training vectors yields better compression rate than when the correlation is less than the training correlation factor.

CHAPTER IX

CONCLUSIONS

Our algorithm performs better than our state-of-the-arts algorithms such as finite-state vector quantization for both speech and image data sets. It works especially well when it is appended by an entropy code such as Huffman coding. This is so because the proportion of re-ordered indices closer to zero is much greater than when indices are obtained from memory-less vector quantization. The likelihood codebook re-ordering algorithm also achieves small coding rates at a smaller computational complexity relative to the Huffman-encoded truncation matrix approach mentioned in chapter 3. The introduction of truncation as mentioned in section 4.4 also shows that the LCR algorithm outperforms DCR and FSVQ for levels of truncation up to 1/8. We also demonstrate in the last chapter the power of learning the transition matrix on an online basis taking in one vector at a time and yielding a smaller entropy as a result of the online training.

REFERENCES

[1] "Pulse code modulation in the telephone industry," *Electrical Engineering*, vol. 77, pp. 769–770, Aug. 1958.

[2] ABATE, J., "Linear and adaptive delta modulation," *Proceedings of the IEEE*, vol. 55, pp. 298–308, March 1967.

[3] ABUT, H., GRAY, R., and REBOLLEDO, G., "Vector quantization of speech and speech-like waveforms," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 30, pp. 423–435, Jun 1982.

[4] ACHARYA, T. and MUKHERJEE, A., "A tree based binary encoding of text using lzw algorithm," in *Data Compression Conference, 1995. DCC '95. Proceedings*, pp. 463–, Mar 1995.

[5] BAKER, R. and GRAY, R., "Differential vector quantization of achromatic imagery," in *Proceedings of International Picture Coding Symposium*, Mar 1983.

[6] BUZO, A., GRAY, A., J., GRAY, R., and MARKEL, J., "Speech coding based upon vector quantization," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '80.*, vol. 5, pp. 15–18, Apr 1980.

[7] BUZO, A., GRAY, A, J., GRAY, R., and MARKEL, J., "Speech coding based upon vector quantization," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, pp. 562–574, Oct 1980.

[8] CHERRY, E. C., "A history of the theory of information," *Proceedings of the IEE - Part III: Radio and Communication Engineering*, vol. 98, pp. 383–393, Sept. 1951.

[9] CHOU, P., LOOKABAUGH, T., and GRAY, R., "Entropy-constrained vector quantization," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, pp. 31–42, Jan 1989.

[10] CHU, C. and ANDERSON, D., "Likelihood codebook reordering vector quantization," in *International Conference on Acoustics, Speech and Signal Processing*, pp. 5114–5117, May 2013.

[11] CUPERMAN, V. and GERSHO, A., "Vector predictive coding of speech at 16 kbits/s," *Communications, IEEE Transactions on*, vol. 33, pp. 685–696, July 1985.

[12] DAVID, E., SCHROEDER, M., LOGAN, B., and PRESTIGIACOMO, A., "Voice-excited vocoders for practical speech bandwidth reduction," *Information Theory, IRE Transactions on*, vol. 8, pp. 101–105, September 1962.

[13] FIORAVANTI, R., FIORAVANTI, S., and GIUSTO, D., "An efficient neural prediction for vector quantization," in *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, vol. v, pp. V/613–V/616 vol.5, Apr 1994.

[14] FIORAVANTI, S. and GIUSTO, D., "A prediction strategy for efficient entropy coding of VQ addresses," in *1994 IEEE International Symposium on Information Theory*, p. 240, July 1994.

[15] FOSTER, J., GRAY, R., and DUNHAM, M., "Finite-state vector quantization for waveform coding," *IEEE Transactions on Information Theory*, vol. 31, pp. 348 – 359, May 1985.

[16] FRANKE, J., "A levinson-durbin recursion for autoregressive-moving average processes," *Biometrika*, vol. 72, no. 3, pp. 573–581, 1985.

[17] GIBSON, J., "Speech coding methods, standards, and applications," *Circuits and Systems Magazine, IEEE*, vol. 5, pp. 30–49, Fourth 2005.

[18] GOODALL, W., "Telephony by pulse code modulation," *Bell System Technical Journal, The*, vol. 26, pp. 395–409, July 1947.

[19] GRAY, R., "Vector quantization," *IEEE ASSP Magazine*, vol. 1, pp. 4 –29, Apr. 1984.

[20] HUFFMAN, D., "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, pp. 1098–1101, Sept 1952.

[21] JAS, A., GHOSH-DASTIDAR, J., NG, M.-E., and TOUBA, N., "An efficient test vector compression scheme using selective huffman coding," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 22, pp. 797–806, June 2003.

[22] JAYANT, N. and NOLL, P., *Digital Coding of Waveforms: Principles and Applications to Speech and Video*. Prentice Hall Professional Technical Reference, 1990.

[23] JUANG, B., WONG, D., and GRAY, A., J., "Distortion performance of vector quantization for LPC voice coding," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 30, pp. 294–304, Apr. 1982.

[24] KOTZE, H. and KUHN, G. J., "An evaluation of the lempel-ziv-welch data compression algorithm," in *Communications and Signal Processing, 1989. COMSIG 1989. Proceedings., Southern African Conference on*, pp. 65–69, Jun 1989.

[25] KOU, W., *Digital Image Compression: Algorithms and Standards*. Norwell, MA, USA: Kluwer Academic Publishers, 1995.

[26] KRISHNAN, V., *A framework for low bit-rate speech coding in noisy environment*. PhD thesis, Mar. 2005.

[27] LANGDON, G.G., J., "An introduction to arithmetic coding," *IBM Journal of Research and Development*, vol. 28, pp. 135–149, March 1984.

[28] LINDE, Y. and A. BUZO, R. G., "An algorithm for vector quantizer design," *IEEE Transactions on Communications*, pp. 84–95, Mar. 1980.

[29] LINDE, Y., BUZO, A., and GRAY, R., "An algorithm for vector quantizer design," *Communications, IEEE Transactions on*, vol. 28, pp. 84–95, Jan 1980.

[30] MEH CHU, C. and PARRISH, N. V., "Adaptive likelihood codebook reordering vector quantization for 1-d data sources," in *IEEE Signal Processing and Signal Processing Education Workshop*, 2015.

[31] NASRABADI, N. and FENG, Y., "Image compression using address-vector quantization," *Communications, IEEE Transactions on*, vol. 38, pp. 2166–2173, Dec. 1990.

[32] NELSON, M. R., "Lzw data compression," *Dr. Dobb's Journal*, vol. 14, no. 10, pp. 29–36, 1989.

[33] O'NEAL, J., J. and WOLF, C., "Quantizer levels for speech differential pcm systems," *Audio and Electroacoustics, IEEE Transactions on*, vol. 18, pp. 315–316, Sep 1970.

[34] O'NEAL, J.B., J. and STROH, R., "Differential pcm for speech and data signals," *Communications, IEEE Transactions on*, vol. 20, pp. 900–912, Oct. 1972.

[35] PAULRAJ, A., ROY, R., and KAILATH, T., "A subspace rotation approach to signal parameter estimation," *Proceedings of the IEEE*, vol. 74, pp. 1044–1046, July 1986.

[36] PENNINGTON, H. and STEELE, R., "A-law pulse-code modulation by a delta-modulation technique," *Electronics Letters*, vol. 9, pp. 171–173, May 1973.

[37] SABIN, M. and GRAY, R., "Product code vector quantizers for waveform and voice coding," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 32, pp. 474–488, Jun 1984.

[38] SAVARI, S., "Redundancy of the lempel-ziv incremental parsing rule," *Information Theory, IEEE Transactions on*, vol. 43, pp. 9–21, Jan 1997.

[39] SCHROEDER, M., "Vocoders: Analysis and synthesis of speech," *Proceedings of the IEEE*, vol. 54, pp. 720–734, May 1966.

[40] SCHROEDER, M. and ATAL, B., "Code-excited linear prediction(celp): High-quality speech at very low bit rates," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '85.*, vol. 10, pp. 937–940, Apr. 1985.

[41] VIRUPAKSHA, K. and O'NEAL, J., "Entropy-coded adaptive differential pulse-code modulation (dpcm) for speech," *Communications, IEEE Transactions on*, vol. 22, pp. 777–787, Jun 1974.

[42] WITTEN, I. H., NEAL, R. M., and CLEARY, J. G., "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, pp. 520–540, June 1987.

[43] YEUNG, R. W., "Rate-distortion theory," in *Information Theory and Network Coding*, pp. 183–210, Springer, 2008.

[44] ZUE, V., SENEFF, S., and GLASS, J., "Speech database development at mit: Timit and beyond," *Speech Communication*, vol. 9, no. 4, pp. 351–356, 1990.