ADAPTIVE QUANTIZATION AND TRANSFORM CODING

by

Jun Zhuang

Memorandum No. UCB/ERL M97/54

4 August 1997

ADAPTIVE QUANTIZATION AND TRANSFORM CODING

by

Jun Zhuang

Memorandum No. UCB/ERL M97/54

4 August 1997

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Acknowledgments

I would like to thank my advisor, Professor Martin Vetterli for his guidance in the past two years. It is Martin who introduced me into this research area and provided me his deep insights and encouraged me with his enthusiasm. I deeply appreciate Martin's commitment to the research and to the career growth of his students.

During my two years study at Berkeley, many friends have given me enormous help. Without their help, I wouldn't have overcome the many difficulties along the way. Among them, I would like to thank all the members of the wavelet group: Vivek Goyal, Joseph Yeh, Grace Chang, Francis Ng, Matt Podolsky, Chris Chan, Steve McCanne and Michael Goodwin. I would like also thank Bin Yu for sharing some of her research results with me. Other friends who have provided supports include Nigel Barboza, Hui Jin, Matthew Longeot, and Patrick Wong. I would also like to thank friends in Pacific Bell: Tom Soon, Jimmy Chuang, and Don Patterson.

I would like thank Professor Venkat Anantharam for being my second reader. Since the day I came to Berkeley, Professor Anantharam has given me so much help, both in terms of academic development and career path. I would like to take this opportunity to thank him. I also thank Mary Byrnes, for her help regarding the graduate matters.

Finally, but not the least, I thank God, for everything that comes from him. He has given me the most blessing I can ever dream of.

Adaptive Quantization and Transform Coding

Abstract

Adaptive Quantization and Transform Coding

by

Jun Zhuang

Master of Science in Engineering-Electrical Engineering and Computer Sciences

University of California at Berkeley

Professor Martin Vetterli, Research Advisor

Universal lossless codes have been proven to exist [22], and practical universal lossless coding schemes have been constructed [2][3][4]. Even though the existence of universal lossy compression algorithms are established in the early 1970's, the development of universal lossy codes remained most in the theoretical domain so far.

The first part of this project is motivated by the work by C. Chan and M. Vetterli [12]. The approach here is to first collect some information about the source based on causal past real samples and then decide whether it is worthwhile to update the coder and send the update as the side information. Unlike in [12], we confine the coder to be a tree structured vector quantizer, which has a low complexity. As a simplified case, we showed how to design such an adaptive coder in the case of scalar quantization. Though similar design can be extended to the vector quantization case, we concluded that though we achieve low complexity by confining the coder to be tree structured, the flexibility to adapt the quantizer suffers as an expense for low complexity, and the side information is increased drastically. Thus adaptive tree structured quantizer remains interesting in theory and after a brief discussion we focus our attention on the more promising backward adaptive Karhunen-Loeve transform coding scheme.

Most the previous work on universal lossy compression are addressed in the vector quantization framework, relatively little work has been done in the transform coding framework. Transform coding techniques are widely used and among various transforms the Karhunen-Loeve transform (KLT) is proven to be the optimal transform under high resolution,

real bit assignment and Gaussian assumptions. In the second part of this project, our objective is to develop a practical backward adaptive Karhunen-Loeve transform coding scheme.

Traditionally, despite its existence as the "optimal" transform, KLT is seldom used in practical transform coding schemes. One of the major difficulties in applying KLT in transform coding is that the KLT is signal dependent and needs to be sent through the communication channels which is usually too expensive. In this project, we use a completely different approach. The essence of this approach is that we measure the autocorrelation of the source signal based on the causal past decoded samples and derive the approximated Karhunen-Loeve transform based on the measurement. Because both the encoder and the decoder have access to these samples, no side information is required. As a further step, we then proved when the source signal can be blocked into i.i.d. Gaussian random vectors, when the step size of the scalar quantizer used is small, the adapted transform converges to the true Karhunen-Loeve transform when both the number of samples used to measure the autocorrelation and the number of adaptation steps go to infinity. In practice, our simulations on AR(1) signal show that for finite number of samples and finite number of adaptations, the adapted transform is very "close" to the true Karhunen-Loeve transform.

Contents

List of Figures

1 Introduction

1.1. Universal Lossy Source Coding

Data compression is about shrinking of data strings before transmission or storage in order to conserve communication or memory costs. In *lossless* compression, it is required that the original string of data can be faithfully reconstructed from the compressed data. In *lossy* compression, controlled fidelity degradation is allowed in the reconstruction in order to achieve higher compression ratio. Typical applications of lossless compression include text compression and certain aspects in multimedia compression. Typical applications of lossy compression include image, video, and speech compression.

In lossy data compression, two techniques are often used, namely, vector quantization (VQ) and transform coding (TC). A brief introduction to transform coding can be found in section 1.2 of this report. Readers who are not familiar with vector quantization are referred to the book "Vector Quantization and Signal Compression" by Gersho and Gray [1].

Lossy data compression plays a vital role in modern communications system. Consider a generic communications system (refer to Fig. 1-1) composed of five elements, a signal source, an encoder, a communication channel, a decoder and a display or output device. The signal source generates at fixed intervals one block or vector chosen among a (continuous or discrete) set S of possible blocks (these blocks are groups of samples of the analog signal that is being transmitted). Call $X_1, \ldots, X_i, \ldots, X_N$ the blocks in the sequence, where X_i will typically denote the current block. The encoder maps each of the source

blocks X_i into a reproduction block $\hat{X}_i$ chosen from a finite codebook $C \subset S$ and transmits the index corresponding to the chosen codeword to the receiver through the communication channel. The decoder maps the index to the reproduction codeword $\hat{X}_i$, which is then presented in the display or output device. Note that we use the terms codebook and index, commonly employed in the vector quantization (VQ) literature, as a conceptual tool, without implying that the encoding procedures or the codebook design are necessarily similar to those used in VQ.

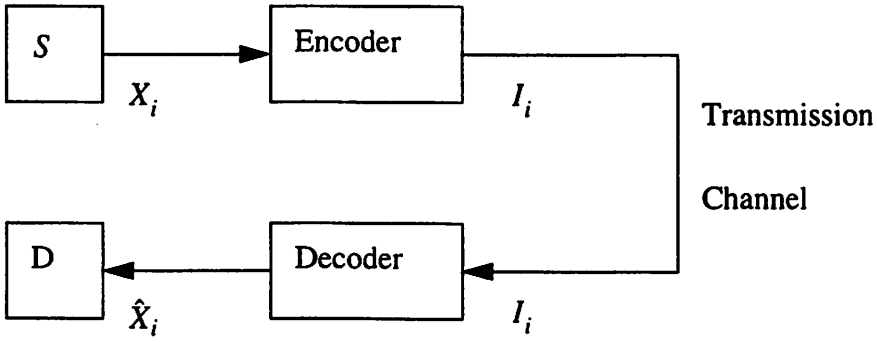

Figure 1-1. Block diagram of a generic communications system.

The fundamental limits to lossy data compression were derived by Shannon in his rate distortion theory. The main theorem [22] of rate distortion theory is that for an *i.i.d.* source X with distribution $p(x)$ and bounded distortion function $d(x, \hat{x})$, there exists a sequence of "optimal" encoding/decoding functions with rate R that achieve a minimum distortion D.

Based on rate distortion theory, practical algorithms have been developed to design encoders/decoders which approximate the "optimal" coder. More or less, these algorithms all share a common feature which is that they either assume a model for the signal source

statistics $p(x)$ or build an empirical model (denote it by $\hat{p}(x)$) based on the observations of the source. When the signal source to be actually coded possesses the same statistics like the model used, the coder designed by these algorithms performs very well.

One such example is the algorithm by Linde, Buzo, and Gray (in this report, we refer to it as the LBG algorithm) which is often used to design the codebook for VQ. During the codebook design stage, a training sequence is formed either by observing the source for a certain period of time, or by drawing samples from a pre-stored database. Then a codebook is designed based on this training sequence and is used to code the source signal. When the source signal has statistics similar to that of the training sequence, good performance is achieved.

Another example is the discrete cosine transform (DCT) used in transform coding. The source signal is assumed to be a first-order Gaussian-Markov process with a large positive correlation coefficient ρ ($\rho \rightarrow 1$). When the source signal actually coded can be reasonably approximated by such a signal, good performance can also be achieved.

However, more often than not, the statistics of the signal source are unknown to the coder and can in practice drastically differ from the model assumed in the designing stage. The LBG algorithm and discrete cosine transform coding (DCT), like other non-adaptive data compression methods, do not perform well when the source statistics of real world data differ from the statistics assumed by the coder. When using a large collection of facial images to train its codebook using the LBG algorithm, a vector quantization coder, for instance, will not perform well on images of landscapes, or brain computer aided tomographies (CATs). Similarly, a discrete cosine transform coder will not perform in coding a text image as well as it does in coding a portrait image.

Therefore, it is natural to ask: without knowing the statistics *a priori*, can we design a coder such that, for a given rate, it can asymptotically achieve the minimum distortion given by Shannon's rate distortion theory? Such encoder/decoder, if they exist, are called *universal* lossy coders and the coding process is called universal lossy source coding.

Universal data compression (source coding) has a substantive literature. In the lossless case, well-known algorithms include the dynamic Huffman algorithm [2], arithmetic coding [3], and the Lempel-Ziv algorithm [4], [5]. These algorithms are not only theoretically interesting, but they have also been found useful in many practical applications. The concept of separating the task of statistical modeling and coding (e.g., Rissanen,[6]) also had an impact on the way people think and use data compression algorithms.

The development of universal lossy data compression remained mostly in the theoretical domain thus far. The works by Gray and others [7] in the early 1970's established the existence of universal lossy compression algorithms. They were later extended to more general source models by many others, e.g. [8], [9].

Currently, there are two typical approaches: The first approach [7] trains a "universal codebook" in advance and makes it available to both the encoder and the decoder. The "universal codebook" consists essentially of the union of optimal codebooks of all sources within consideration. if the number of sources is not excessive, the "overhead" for taking the union is asymptotically negligible. The second approach does not train a codebook in advance. The encoder constructs a codebook base on the observation of a sufficiently long prefix of source symbols, and transmits the codebook to the decoder. The algorithms of Ornstein and Shields [9], and Ziv [10],[11] are such examples.

In [12] Chan and Vetterli proposed an algorithm called *rate-distortion Lempel-Ziv (RDLZ)* algorithm using the framework of adaptive vector quantization. In their algorithm, without knowing the statistics of the source, the coder starts with an initial codebook and update the codebook according to the statistics of the source during the coding process. A source vector can be simply encoded by a codevector in the current codebook, or it can invoke the addition of a new codevector. Current codevectors can be moved around or even deleted. These actions are taken to minimize the Lagrangian $R + \lambda D$, where R is the bit rate involved in taking an action, D is the distortion it introduces, and λ is a parameter chosen to control the operating point of the algorithm on the operational rate distortion curve. In a sense, bits are "traded" for a reduction in distortion by modifying the codebook.

In the adaptive scalar quantization case, Ortega and Vetterli [13] proposed a model-based adaptive scalar quantization algorithm based on the quantized causal past data with convincing simulation results. They divide the adaptive quantization problem based on quantized past data into two parts, namely, model estimation and quantizer design. Without knowing the statistics of the source, the adaptive quantizer starts with a uniform quantizer. During the quantization process, the model estimation produces an estimate of the source probability density function based on the causal past quantized data, which is used to re-design the quantizer using standard techniques such as Lloyd-Max technique.

The first part of this project is motivated by the work by C. Chan and M. Vetterli [12]. The approach is to first collect some information about the source based on causal past samples and then decide whether to update the coder and send the update as side information. Unlike [12], we confine the coder to be a tree structured vector quantizer, which has a low complexity. As a simplified case, we showed how to design such an adaptive coder in the case of scalar quantization. Though similar design can be extended to the vector quantization case, we concluded that though we achieve low complexity by confining the coder to be tree structured, the flexibility to adapt the quantizer suffers as an expense for low complexity, and the side information is increased drastically. Thus adaptive tree structured quantization remains interesting in theory and in this report we only included a brief discussion illustrating the ideas how an adaptive tree structured vector quantization scheme may be designed and will leave the design and experimental results to the future work.

However, most of the previous work on universal lossy compression are done in the vector quantization framework. So far, little work on universal lossy compression has been done in the transform coding framework. Transform coding is widely used in current image and video compression standards [14],[15]. The second part of this project is motivated by addressing universal lossy compression in the transform coding framework. Specifically, we present an adaptive Karhunen-Loeve transform coding algorithm.

1.2. Transform Coding and Karhunen-Loeve Transform

Suppose that we have a block of consecutive samples of a stationary random process, and we wish to efficiently code this block with a specified number of bits. Let X denote the sample vector $X = (X_1, X_2, \ldots, X_k)^t$. These samples will typically have substantial correlation and separate quantization of each would be an inefficient way to encode them when the quota of bits is relatively modest. The idea of transform coding is that by performing a suitable linear transformation on the input vector X, we can obtain a new vector Y, also with k components, often called transform coefficients or simply, coefficients, with the feature that these coefficients are much less correlated than the original samples. In addition, the information may be more "compact" in the sense of being concentrated in only a few of the transform coefficients. Having in this sense removed redundancy, we hope as a result to be able to quantize these components more efficiently. Fig. 3-2 illustrates the structure of transform coding where T is a k by k invertible matrix that performs the linear transformation.

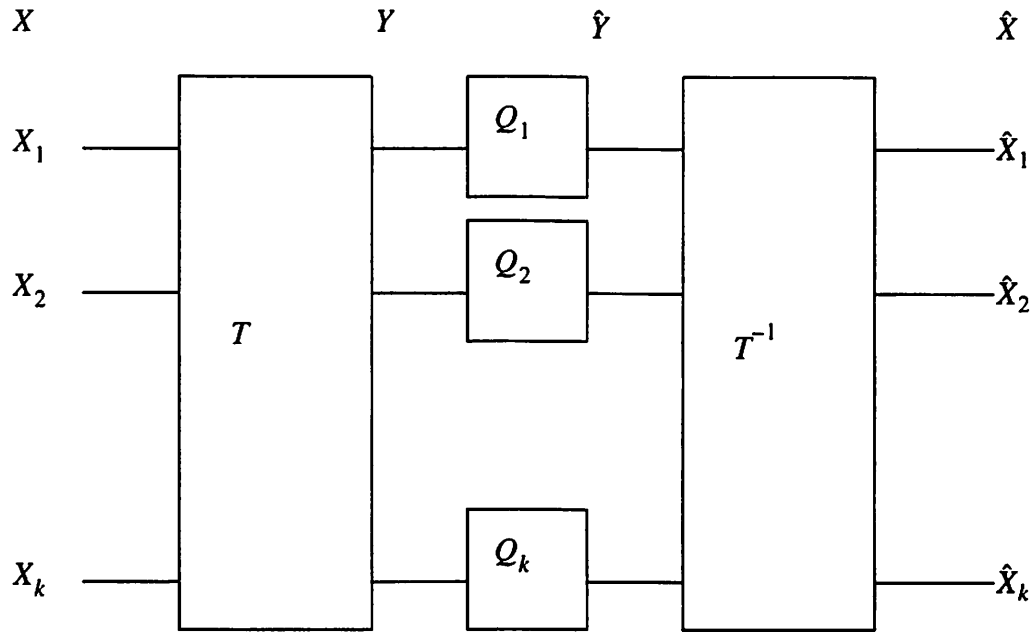

Figure 1-2. Transform coding

In general, the components of X are correlated with one another. However, it is indeed possible to select an orthogonal matrix T, for a given p.d.f. describing X, that will make $Y = TX$ have pairwise uncorrelated components. This choice of transform matrix that makes the linear transformation have this desirable matrix is called discrete time Karhunen-Loeve transform and is defined below.

Let $R_X = E[XX^t]$, denote the autocorrelation matrix of the input vector X (assume $E[X] = 0$). Let u_i denote the eigenvectors of R_X (normalized to unit form) and λ_i the corresponding eigenvalues. Since any autocorrelation matrix is symmetric and nonnegative definite, there are k orthogonal eigenvectors and the corresponding eigenvalues are real and nonnegative. Without loss of generality we assume the indexing is such that

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k \geq 0$$

The Karhunen-Loeve transform matrix can be defined as $T = U^t$, where

$$U = [u_1, u_2, \ldots, u_k]$$

that is, the columns of U are the eigenvectors of R_X.

Then the autocorrelation matrix of Y is given by

$$R_Y = E[YY^t] = E[U^tXX^tU] = U^tR_XU = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & \lambda_k \end{bmatrix}$$

Thus we can see that the Karhunen-Loeve transform does indeed decorrelate the input vector. It also follows that the variances of the transform coefficients are the eigenvalues of the autocorrelation matrix of R_X.

The optimality of Karhunen-Loeve Transform (KLT) for transform coding of a stationary Gaussian source under high resolution and arbitrary real bit assignment assumptions is well known [1]. Adaptive versions of KLT coding are not widely used because in traditional adaptive systems (with side information) transmission of the KLT coefficients can be prohibitively expensive. In Chapter 3 we propose a backward adaptive Karhunen-Loeve transform coding scheme which can adapt to the source statistics without transmitting side information.

1.3. Outline of the Report

In chapter 2, section 2.1 describes adaptive tree structured scalar quantization (TSSQ). Adaptive tree structured vector quantization (TSVQ) is discussed in section 2.2.

Chapter 3 describes adaptive transform coding algorithm. Section 3.2 gives two theorems that show in theory backward adaptive Karhunen-Loeve transform coding exists. Then section 3.3 presents a practical backward adaptive KLT coding algorithm and gives a theoretical analysis of the algorithm by proving two theorems which shows the universality of the adaptive algorithm. In section 3.4, practical issues are considered. Finally, experimental results are presented in section 3.5.

Chapter 4 gives the conclusions, and indicates future work.

2 Adaptive TSSQ and Adaptive TSVQ

2.1. Introduction

In [12] Chan and Vetterli proposed an adaptive lossy data compression technique called rate distortion Lempel-Ziv (RDLZ) using vector quantization (VQ) framework. In their proposed algorithm, for each source vector, a full search through the VQ codebook is used during the encoding process. The drawback of the full search technique is that its complexity, especially when the vector size is large, is much higher than that of other technique such as tree structured VQ (TSVQ), which has a much reduced complexity at a small expenses of performance degradation. Thus, it is natural to ask: can we design a RDLZ technique for TSVQ? The work presented in this chapter is motivated by this question.

Vector quantization is a generalization of scalar quantization. The task of designing a quantization technique in the scalar case is, in general, easier than in the vector case. For this reason, we first focus our attention on designing a RDLZ technique for tree structured scalar quantizer (TSSQ).

2.2. Adaptive TSSQ

Let $x = (x_1, x_2, \ldots)$, $x_i \in R$, be a realization of a random vector X with probability density function $f(x)$ and cumulative distribution function $F(x)$, which are unknown to the quantizer. Prior to the quantization process, the quantizer is set to be some initial quantizer.

We group data strings $x_1, x_2, \ldots$ into blocks of size L and at the end of each block, the statistics of the source($F(x)$, for example) are estimated based on the causal past m blocks of *unquantized* data. The quantizer is then updated and is used to quantize the next block.

Denote the data blocks by $\underline{x}_j = (x_{(j-1)L+1}, \ldots, x_{jL})$ and its quantized counterpart by $\hat{\underline{x}}_j = (\hat{x}_{(j-1)L+1}, \ldots, \hat{x}_{jL})$. Then at the end of the j-th data block, $\hat{F}_j = \hat{F}_j(\underline{x}_j, \ldots, \underline{x}_{j-max(1,m)+1})$. The quantizer for the next data block is, therefore, $\hat{Q}_j = \hat{Q}_j(\underline{x}_j, \ldots, \underline{x}_{j-max(1,m)+1})$.

In practice, when we update the quantizer for the j-th block, we need to inform the decoder of this update in the form of side information, which, in terms of the trade-off between bit rate and distortion, is not always a wise thing to do. Thus, we need a criterion to decide when to update the quantizer and when not to.

Thus, we can see that our adaptive quantizer should at least have three parts:

1. Model estimation: Based on previous N *unquantized* samples we estimate the distribution function (we decided on cumulative distribution function) of the source.

2. Quantizer design: for a given estimated c.d.f. design a new tree-structured scalar quantizer.

3. Adaptation decision: decide whether or not to adapt the quantizer based on the trade-off between bit rate and distortion.

The following sections discuss, one by one, these three parts.

2.2.1. Estimation of Source Statistics

For this part, we want to measure the cumulative distribution function $F(x)$ of the source based on the causal past *unquantized* samples. We set our objective as the following:

Objective 1: Given the N most recent unquantized sample occurrence $x(n-N), x(n-N+1), \ldots, x(n-1)$, where N might be a constant or can be changed by the speed of adaptation algorithm, find an estimate $\hat{F}(x)$ of the cumulative distribution function (c.d.f.) of the source, $F(x)$.

We define, for $\forall x \in R$, $n(x)$ the count of samples falling in the region $(-\infty, x)$. That is $n(x) = \sum_{i=1}^{N} I_{\{X_i < x\}}$.

Let $\hat{F}(x) = \frac{n(x)}{N}$, then as $N \to \infty$, $\hat{F}(x)$ is a consistent estimate of $F(x)$.

2.2.2. Tree Structured Scalar Quantizer Design

For this part, we set our objective as the following:

Objective 2: For a given c.d.f. $F(x)$, design a tree structured scalar quantizer.

Before we give the procedure to design a tree structured scalar quantizer, we first prove the following proposition.

Proposition 2-1: Given a random variable X taking value on real set R and having $f(x)$ and $F(x)$, respectively, as its density function and cumulative density function. Then for a given set $[a, b]$, the conditional expectation $E[X|a \le X < b]$ is:

$$E[X|a \le X < b] = \frac{F(b)\,b - F(a)\,a}{F(b) - F(a)} - \frac{1}{F(b) - F(a)} \int_a^b F(x)\,dx.$$

Proof:

$$E[X|a \le X < b] = \frac{1}{F(b) - F(a)} \int_a^b x f(x)\,dx = \frac{1}{F(b) - F(a)} \int_a^b x\,dF(x)$$

$$= \frac{1}{F(b) - F(a)} \left[xF(x) \Big|_a^b - \int_a^b F(x)\,dx \right]$$

$$= \frac{F(b)\,b - F(a)\,a}{F(b) - F(a)} - \frac{1}{F(b) - F(a)} \int_a^b F(x)\,dx \quad \square$$

There are several special cases:

Special case (1): $a = -\infty$, $b < \infty$

$$E[X | a \le X < b] = b - \frac{1}{F(b)} \int_{-\infty}^{b} F(x)\,dx$$

Special case (2): $a > -\infty$, $b = \infty$

$$E[X | a \le X < b] = \frac{1}{1 - F(a)} \int_a^{\infty} (1 - F(x))\,dx$$

Special case (3): $a = -\infty$, $b = \infty$

$$E[X | a \le X < b] = \int_0^{\infty} (1 - F(x))\,dx - \int_{-\infty}^{0} F(x)\,dx$$

One may argue that in the special cases, the integration may not exits. However, because in our application the empirical cumulative distribution function is always measured based on finite number of source samples, this empirical cumulative distribution function is always integrable.

For a random variable X which has c.d.f. $F(x)$, the tree-structured scalar quantizer is designed according to the following procedure. In order to simplify the notations, we give the design procedure for a binary tree quantizer with depth of two (refer to Fig. 2-1).

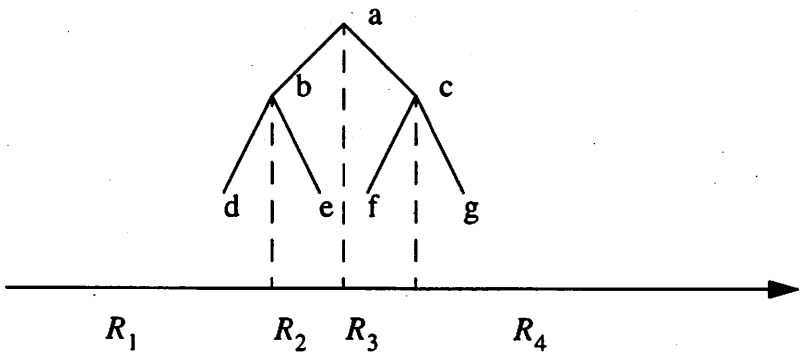

Figure 2-1. Tree structured scalar quantizer design

From Fig. 2-1 we can see that a divides the real line into two regions. Denote these two regions by $R_1 \cup R_2$ and $R_3 \cup R_4$, respectively. b, the left child of a divides the region $R_1 \cup R_2$ into two regions, R_1 and R_2. c, the right child of a divides the region $R_3 \cup R_4$ into two regions, R_3 and R_4. The reconstructed values of region R_1, R_2, R_3, R_4 are d, e, f and g, respectively.

Then if the c.d.f. of the source is $F(x)$, then we determine the values of $a, b, \ldots, g$ in the following way:

$$a = E[X|X \in -(\infty, \infty)]$$

$$b = E[X|X \in R_1 \cup R_2]$$

$$c = E[X|X \in R_3 \cup R_4]$$

$$d = E[X|X \in R_1]$$

$$e = E[X|X \in R_2]$$

$$f = E[X|X \in R_3]$$

$$g = E[X|X \in R_4]$$

From Proposition 2-1, we already know how to calculate the conditional expectation using the c.d.f. $F(x)$.

If we want to design a tree structured quantizer with depth larger than two, we can design it in a very similar way.

2.2.3. Adaptation Criterion

When we update the quantizer, we need to send update information to the decoder. Updating the quantizer reduces the distortion, but at the same time costs extra bits. Thus there is a trade-off between the bit rate and the distortion. Thus we need a criterion to decide when to send side information and when not to. This can be formulated as minimizing a cost function, which in the current context is the Lagrangian $J = R + \lambda D$.

Given the current c.d.f $F(x)$, the current quantizer $Q_{current}$ and the candidate quantizer $Q_{candidate}$, we can compute the additional bits needed to update the quantizer ΔR, and the reduced distortion ΔD if we replace $Q_{current}$ with $Q_{candidate}$. For a given parameter λ, we then calculate $\Delta J = \Delta R + \lambda \Delta D$. If $\Delta J > 0$, the quantizer $Q_{current}$ is replaced with $Q_{candidate}$. Otherwise, keep the current quantizer $Q_{current}$ unchanged.

2.3. Adaptive TSVQ

Having presented an adaptive quantization scheme for the scalar case, we now look at adaptive quantization in the general vector case. Similarly, an adaptive TSVQ algorithm should at least have three parts:

1. Model estimation: Estimate the source statistics based on previous N *unquantized* samples.

2. Quantizer design: Based on the estimated source statistics, design a new tree-structured vector quantizer.

3. Adaptation decision: decide whether or not to adapt the TSVQ quantizer.

2.3.1. Estimation of Source Statistics

Unlike in the scalar case, there is no obvious way to estimate the source statistics in terms of its probability density function or its cumulative density function. In [12], Chan uses the following method to estimate the source statistics: Denote the current codebook by $C_{current}$. The codebook is a set of reproduction k-dimensional codevectors $\{c_j \in R^k, j = 1, \ldots, M\}$, each of which is associated with a cell $R_j = \{x_i | \hat{x}_i = c_j\}$, where $\hat{x}_i$ is the quantized version of x_i, the source vector. Among the recent previous N source vectors, we record the counts of the source vectors that fall into each codevector cell. Denote this by n_j, the occurrence count for the j-th codevector among the N source vectors. Then $n_j, j = 1, \ldots, M$, give us some information on the source statistics.

In the TSVQ case, we propose a similar method to estimate the source statistics. In order to simplify the notations, we consider a TSVQ with depth of two (refer to Fig. 2-2).

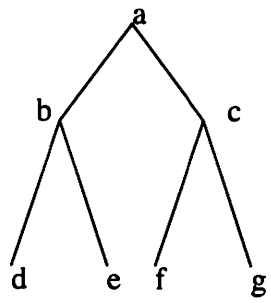

Figure 2-2. An example of TSVQ

$a, b, \ldots, g$ are k-dimensional vectors. $a = E[X|X \in R^k]$. b and c divides vector space R^k into two regions, R_1 and R_2 respectively. d and e divides R_1 into R_3 and R_4. f and g divides R_2 into R_5 and R_6. d, e, f and g are also the reconstructed codevectors

of region R_3, R_4, R_5 and R_6, respectively. Among the N recent source vectors, we count the number of vectors falling into each region and denote them by $n_1, n_2, \ldots, n_6$. Then we know that $\sum_{i=3}^{6} n_i = N$, $n_1 = n_3 + n_4$ and $n_2 = n_5 + n_6$. Then n_i, $i = 1, 2, \ldots, 6$ give us some information about the source statistics.

2.3.2. Tree-Structured Vector Quantizer Design

As we have mentioned before, TSVQ imposes a strong constraint on the structure of the codebook. The codevectors in the codebook are no longer independent of each other as they are in the full search VQ case. As a result, new codevectors can no longer be handily added or deleted. This is a disadvantage we have to face when dealing with structured codebook. Nonetheless, we can still update the codevectors. Take the depth-two TSVQ in the previous section as an example. There are n_3 vectors falling into region R_3. Then we can calculate the centroid of these n_3 vectors and replace the old codevector with this new centroid. There are $n_1 = n_3 + n_4$ vectors falling into region R_1, so we can calculate the centroid of these n_1 vectors and use it as the new codevector for region R_1.

Generally speaking, if among the N source vectors there are n_j vectors fall into the region R_j associated with the codevector c_j, then we can move the codevector c_j to the centroid of these n_j vectors for the next data block.

2.3.3. Adaptation Criterion

Again, when we update the current codebook, we need to send the update information to the decoder. So we need a criterion to decide when to update the codebook and when not to. A method very similar to the one mentioned in scalar case can be used here. when the Lagrangian $J = R + \lambda D$ is minimized, then update the quantizer. Otherwise, keep the quantizer unchanged

One important feature of TSVQ is that it is not only a codebook which is fast to search but also a multi-resolution codebook. If we want to maintain this multi-resolution feature, we need to send the update information of all the levels of the codebook to the decoder. If we do not need to maintain this multi-resolution feature, we need only send the update information about the bottom nodes of the tree to the decoder.

2.4. Comments and Future Directions

Adaptive vector quantization has been intensively investigated by researchers in recent years. Chan and Vetterli [12] proposed a vector quantization compression scheme with the goal to design an adaptive Lempel-Ziv coding scheme in the lossy compression domain. Encouraged by their results [12], we began to focus on reducing the complexity of the compression scheme. So we set our objective as to design an adaptive tree structured vector quantization scheme which can adapt to the source statistics in a way similar to the Lemple-Ziv algorithm.

As a case study, we studied tree structured scalar quantization and found that in the scalar case, the source cumulative distribution function can be measured based on causal unquantized samples, and the tree structured codebook can be designed based on this measurement. Trade-off between side information and the distortion can be evaluated based on minimizing the Lagrangian.

3 Adaptive KLT without Side Information

3.1. Introduction

The optimality of the Karhunen-Loeve Transform (KLT) for transform coding of a stationary source under high resolution and arbitrary real bit assignment assumptions is well known [16]. However, the KLT transform is signal dependent. This is both an advantage and a disadvantage: The advantage is that being signal dependent, KLT transform coding inherently is an adaptive coding scheme. The disadvantage is that the KLT transform must be sent to the decoder during the coding process and in most applications it is prohibitively expensive to send this side information.

In order to apply Karhunen-Loeve Transform in data compression, two approaches have been investigated. The first approach is based on classification methods [23]. The signal space is divided into a finite set of classes and a fixed Karhunen-Loeve transform is designed for each class. In the coding process, the transform is switched between the collection of transforms to adapt to the local statistics of the source. The second approach is based on the on-line adaptation algorithm. It can be described as following:

Let $\{X_t | t = 1, 2, \ldots\}$ be a stationary random process and is blocked into $(X_1, \ldots, X_L)\ (X_{L+1}, \ldots, X_{3L})\ (X_{3L+1}, \ldots, X_{7L}) \ldots$, where the blocks have size of $L, 2L, 4L, 8L, 16L, \ldots$ respectively. The statistics of the random process is measured for each block and the Karhunen-Loeve transform computed and is sent as side information

through the communication channel. When $\{X_t | t = 1, 2, \ldots\}$ is stationary and L is sufficiently large, the computed transform will converge to the true Karhunen-Loeve transform and the number of bits needed to send the side information will become negligible when the size of the block goes to infinity.

The above scheme has two drawbacks. First, when the block size becomes large, it is difficult to estimate the source statistics for the block. Second, the above scheme is effective only when the source is stationary. In the case of a non-stationary source, statistics measured for large size block will not represent the local source statistics well.

In [13], Ortega and Vetterli proposed an adaptive scalar quantization scheme using backward adaptation and achieved very good simulation results. Because the adaptation is based on the causal past quantized data, no side information is needed. It is natural to ask: is there a backward adaptive Karhunen-Loeve transform coding scheme and how well does it perform? Motived by the question and in collaboration with Goyal, we propose in this report a backward adaptive Karhunen-Loeve transform coding scheme (Figure 3-1).

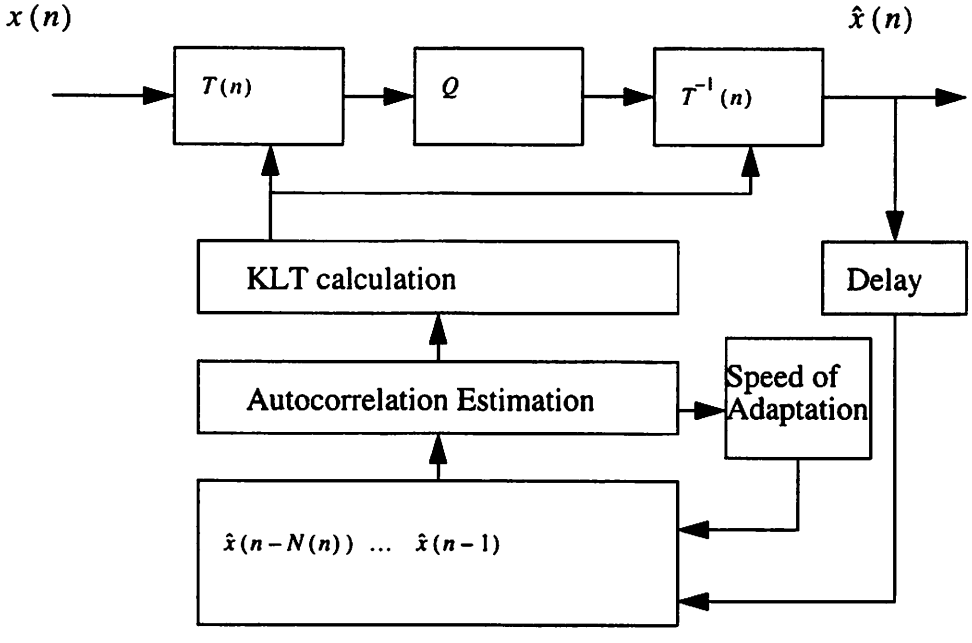

Figure 3-1. Backward adaptive Karhunen-Loeve transform coding scheme

3.2. Preliminary Discussions

3.2.1. Backward Adaptation

We consider a stationary random processes $X(\theta) = \{X_t | t = 1, 2, \ldots\}$ depending on a vector $\theta = (\theta_1, \ldots, \theta_k)$ of real-valued parameters and let θ^* be an estimator of θ.

We group $X_1, X_2, \ldots$ into blocks of size L and at the end of each block, θ is estimated based on m blocks of the causal past *quantized* data, and the coder, denoted by Q_j, is updated based on the estimation of θ.

Denote the data blocks by $\underline{X}_j = (X_{(j-1)L+1}, \ldots, X_{jL})$ and its quantized counterpart by $\hat{\underline{X}}_j = (\hat{X}_{(j-1)L+1}, \ldots, \hat{X}_{jL})$. Then at the j-th data block: $\theta^*_j = \theta^*_j(\hat{\underline{X}}_j, \ldots, \hat{\underline{X}}_{j-max(1,m)+1})$. The quantizer for the next data block is, therefore, $\hat{Q}_j = \hat{Q}_j(\theta^*_j)$.

3.2.2. Parameters Measurement Based on Quantized Samples

In this section, we will show that backward adaptive KLT transform coding scheme exists. In order to simplify the notations, we present this section in the case of dimension two without losing any generality.

Let $(X_1, X_2, \ldots)$ be *i.i.d.* random vectors. X_i is of the form:$X_i = \left[X_i^{(1)}, X_i^{(2)}\right]^t$, where $X_i^{(1)}$ and $X_i^{(2)}$ take values from the real set $[a, b]$. The joint probability density function of $X_i^{(1)}, X_i^{(2)}$ is $f\left(x^{(1)}x^{(2)}|\theta\right)$, where $\theta \in R^M$ is the parameter vector.

let $Q = (Q_1, Q_2)$ be an two dimensional quantizer, where Q_1 and Q_2 are, respectively, level L_1 and level L_2 scalar quantizers of the real set $[a, b]$. Let

$R_i = \{a = b_0^{(i)} < b_1^{(i)} < b_2^{(i)} < \ldots < b_{L_i}^{(i)} = b\}$, $i = 1, 2$, be an L_i-level internal partition of $[a, b]$ and $C_i = \{c^{(i)}(j) | j = 1, 2, \ldots, L_i\}$ is the reconstruction level for Q_i.

Assume we only observe the quantized causal past N data vectors. Denote by $n_{ij}(N)$ the counts of X's falling into intervals of the partition $R_i \times R_j$, $i, j = 1, 2$:

$$n_{ij}(N) = \sum_{m=1}^{N} I_{\{b_{i-1}^{(1)} < X_m^{(1)} \le b_i^{(1)}\}} I_{\{b_{j-1}^{(2)} < X_m^{(2)} \le b_j^{(2)}\}}$$

Assume the input source process is stationary with a M dimensional parametric marginal density $f\left(x^{(1)} x^{(2)} | \theta\right)$ $\left(\theta \in R^M\right)$. For $i = 1, 2, \ldots, L_1$, $j = 1, 2, \ldots, L_2$ denote

$$p_{ij} = \int_{b_{i-1}^{(1)}}^{b_i^{(1)}} \int_{b_{j-1}^{(2)}}^{b_j^{(2)}} f\left(x^{(1)}, x^{(2)} | \theta\right) dx^{(1)} dx^{(2)} \qquad (3\text{-}1)$$

where

$i = 1, 2, \ldots, L_1$, and $j = 1, 2, \ldots, L_2$.

Note that $\sum_{i=1}^{L_1} \sum_{j=1}^{L_2} p_{ij} = 1$.

Let $L = \prod_{i=1}^{2} L_i$. Then vector $P = (p_{ij})$, $i = 1, \ldots, L_1$, $j = 1, \ldots, L_2$ is a L-dimensional vector.

Proposition 1 Assume $(X_1, X_2, \ldots)$ are *i.i.d.* random vectors, and $L = M + 1$. If matched with a probability vector $P = (p_{ij})$, equations in (3-1) have a unique solution

$\hat{\theta} = g(p_{ij})$ and g is a continuous function, then $f(x|\theta)$ can be estimated consistently based on quantized data $n_{ij}(N)$.

Proof: For $L = M + 1$, any $L - 1$ of the equations in (3-1) uniquely determine θ in terms of p_{ij}'s: $\theta = g(p_{ij})$. By ergodicity, when $N \to \infty$, $n_{ij} \to p_{ij}$; hence $\hat{\theta} = g(n_{11}/N, \ldots, n_{L_1 L_2}/N)$ tends to the true θ as N gets large because g is continuous. □

Proposition 1 is actually an extension of the theorem 1 in [24]. Yu proved that under certain conditions, parameters of stationary and ergodic one-dimensional random variables can be consistently measured based on quantized samples. Here, we propose that this is also true for multi-dimensional random variables.

To put things into perspective, consider the following special case:

Let $X_i = \left[X_1^{(i)}, X_2^{(i)}\right]^t$ be a two dimensional Gaussian random vector with the autocorrelation matrix $\begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$ and mean $[\mu_1, \mu_2]^t$. Then it's distribution is determined by parameter vector $\theta = [\sigma_1, \sigma_2, \rho, \mu_1, \mu_2]^t$. If equations in (3-1) has unique solution, then it is possible to solve for θ based only on n_{ij}.

The next question is in the above special case, do equations in (3-1) have unique solutions?

First, it is easy to verify that $\sigma_1, \sigma_2, \mu_1, \mu_2$ have unique solutions (refer to [24]). So ρ is the only parameter need to be determined. For the purpose of brevity, we leave the discussion to the Appendix A.

Proposition 2 Let X_i be zero mean k-dimensional Gaussian random vector with autocorrelation matrix R_X. Let T be an orthonormal transform and let Q be a k-dimensional scalar quantizer with $L = \prod_{i=1}^{k} L_i > \frac{k(k+1)}{2}$ reconstruction levels. Then the KLT transform can be consistently reconstructed based on quantized random vector $\hat{X}_i$.

Proof: Let $Y_i = TX_i$. Then Y_i is also a Gaussian random vector with autocorrelation matrix $R_Y = TR_XT^t$. Thus $R_X = T^tR_YT$. From theorem 1, we know that based on the quantized data $\hat{Y}_i$, the parameters of Y_i (in this case, R_Y) can be reconstructed. Therefore, the parameters of X_i can also be computed and so does the KLT transform, which is the transpose of the eigenvector matrix of R_X. □

Theorem 2 implies that if we have an initial transform T and a k-dimensional fixed-rate quantizer Q formed by k scalar quantizers, for a Gaussian vector source with autocorrelation R, we can consistently estimate R based on the causal past quantized data. Then the true Karhunen-Loeve transform can be derived by finding the eigen-decomposition of R. Thus, backward adaptive KLT transform coding scheme, in theory, exists.

3.3. Backward Adaptive Karhunen-Loeve Transform Coding

Theorem 1 and 2 in the previous section have proved that backward adaptive Karhunen-Loeve transform coding exists in the parametric case. However, the method in theorem 1 and 2 can not be implemented in a straightforward way; It requires solving nonlinear equations, which is a non-trivial task. In this section, we will propose a practical backward adaptive Karhunen-Loeve transform coding scheme (Fig.3-1).

3.3.1. Quantizer Model

In our adaptive Karhunen-Loeve transform coding scheme, instead of using a fixed rate quantizer as we did in section 3.2, we use a subclass of uniform quantizers that can be

described by a two-parameter models used. The operation of a representative quantizer of this subclass is governed by the following general two-parameter (a, Δ) model:

$$X_Q = \left(a + n + \frac{1}{2}\right)\Delta,$$

if $(a+n)\Delta \le X < (a+n+1)\Delta$ with $a \in \left[-\frac{1}{2}, \frac{1}{2}\right)$ and $n \in Z$ where X, X_Q, Δ and a are the unquantized and quantized variables, the uniform step size and shift factor, respectively. It is clear from this model that the two most important quantizers, namely the mid-stepper (with dead zone) and the mid-riser (without dead zone) are characterized by $a = -\frac{1}{2}$ and $a = 0$, respectively.

3.3.2. Overview of the Algorithm

Let $x = (x_1, x_2, \ldots)$, $x_i \in R^k$, be a realization of the k dimensional random vector X with the autocorrelation matrix $R_X = E[XX^T]$. For an orthonormal transform T, the output of the transform block, $y = (y_1, y_2, \ldots)$, where $y_i = Tx_i$, is a realization of a random vector Y with autocorrelation matrix R_Y. The quantizer Q consists k uniform quantizers Q_i with step size $\Delta_i, i = 1, 2, \ldots, k$. The output of the quantizer, $\hat{y} = (\hat{y}_1, \hat{y}_2, \ldots)$, where $\hat{y}_i = Q(y_i)$ is a realization of the k dimensional random vector $\hat{Y}$ with the autocorrelation matrix $R_{\hat{Y}}$. Let $\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots)$, where $\hat{x}_i = T^t \hat{y}_i$, be the output of the inverse transform block. Then $\hat{x}$ is a realization of random vector $\hat{X}$ with autocorrelation matrix $R_{\hat{X}}$.

Encoding the data string $x = (x_1, x_2, \ldots)$ using a fixed transform gives an operating point on the R-D plane. We want to adapt the transform so that the operating point moves towards the R-D bound for the source. We also want to adapt the transform to the local statistics of the source. In order to achieve adaptivity, the data string $x_1, x_2, \ldots$ is

grouped into blocks of size L and at end of each block, the autocorrelation matrix of $\hat{X}$ is estimated based on the causal past m blocks of quantized data. The transform is then updated and is used to quantize the next block.

Denote the data blocks by $\underline{x}_j = (x_{(j-1)L+1}, \ldots, x_{jL})$ and its quantized counterpart by $\hat{\underline{x}}_j = (\hat{x}_{(j-1)L+1}, \ldots, \hat{x}_{jL})$. Then at the end of the j-th data block, $R_{\hat{X}}^*{}_j = R_{\hat{X}}^*{}_j(\hat{\underline{x}}_j, \ldots, \hat{\underline{x}}_{j-max(1,m)+1})$. The transform for the next data block is, therefore, $T_j = T_j(\hat{\underline{x}}_j, \ldots, \hat{\underline{x}}_{j-max(1,m)+1})$.

As an overview, the algorithm goes as follows:

1. Start with an initial transform T_1. Let $j = 1$.

2. For the j th data block, on each source vector x, apply the orthonormal transform T_j to obtain the coefficient vector y.

3. Apply the uniform quantizer Q on y to obtain the quantized coefficient vector $\hat{y}$.

4. Apply the inverse transform T_j^t to obtain the reconstructed vector $\hat{x}$.

5. At the end of j th data block, estimate the autocorrelation matrix $R_{\hat{X}}$ based on the causal past m blocks quantized data and compute the new transform T_{j+1} by finding the eigen-decomposition of $R_{\hat{X}}$.

6. Let $j = j + 1$ and go to step 2.

3.4. Analysis

In this section, we will show that the algorithm we proposed in section 3.3 achieves a certain "*universality*" for a source with unknown statistics. We assume the source statistics is characterized by its autocorrelation matrix R_X which is unknown to the coder. Ini-

tially, the transform coder starts with an initial transform T_0. Rewrite the six-step adaptation scheme in section 3.3 in terms of the autocorrelation matrix, we get:

Step 1. For a given R_X and transform T_n, initially $n = 1$, $R_Y^{(n)} = T_n R_X T_n^t$

Step 2. After the uniform quantizers, $R_{\hat{Y}}^{(n)}$, the auto-correlation matrix of $\hat{Y}$, is a function of $R_Y^{(n)}$ and the step-size of the quantizers, Δ. Let's denote it by $R_{\hat{Y}}^{(n)} = C_\Delta\left(R_Y^{(n)}\right)$.

Step 3. The reconstructed vector $\hat{X}$ has an auto-correlation matrix $R_{\hat{X}}^{(n)}$, which is $T_n^t R_{\hat{Y}}^{(n)} T_n$. We perform an eigen-decomposition on $R_{\hat{X}}^{(n)}$ so that $R_{\hat{X}}^{(n)} = T_{n+1}^t D_n T_{n+1}$, where T_{n+1} is an orthonormal transform and D_n is a diagonal matrix.

Step 4. Let $n = n + 1$ and go to step 1.

In section 3,4.2 we will analyze the behavior of the algorithm, but first we need to find out the effects of uniform quantizer on the autocorrelation matrix. More specifically, we need to find out the function $R_{\hat{Y}} = C_\Delta(R_Y)$.

3.4.1. Effect of Quantizer on Autocorrelation Matrix

In [16] Cheded studied the following question:

Given a k-dimensional unquantized (or input) vector signal $X = [X_1, \ldots, X_k]^t$ which is quantized by a k-dimensional uniform quantizer Q of step size Δ into a k-dimensional quantized (or output) vector $X_Q = [X_{Q1}, \ldots, X_{Qk}]^t$, what is the exact relationship

between the $(p_1, \ldots, p_k)$ -th unquantized moment, $E\left[\prod_{j=1}^{k} X_j^{p_j}\right]$, and its quantized counterpart, $E\left[\prod_{j=1}^{k} X_{Qj}^{p_j}\right]$?

The exact solution to this k-dimensional moments relationship problem can be summarized as follows: if $W(u_1, u_2, \ldots, u_k)$ and $W_Q(u_1, u_2, \ldots, u_k)$ are, respectively, the input and output characteristic functions (CFs) of the k-dimensional quantizer Q whose step size is Δ_i, $i = 1, 2, \ldots, k$, then these two functions can be shown[17] to be related by the following input/output CF equation:

$$W_Q(u_1, u_2, \ldots, u_k) = \sum_{n_1, \ldots, n_m \in Z^k} \exp\left(-i2\pi \sum_{j=1}^{k} \left(a_j + \frac{1}{2}\right) n_j\right) W\left(u_1 + \frac{2n_1\pi}{\Delta_1}, \ldots, u_k + \frac{2n_k\pi}{\Delta_k}\right)$$

$$\times \prod_{j=1}^{k} \frac{\sin\left(\frac{\Delta_j u_j}{2} + n_j\pi\right)}{\left(\frac{\Delta_j u_j}{2} + n_j\pi\right)}, \qquad (3\text{-}2)$$

where $i = \sqrt{-1}$.

The k-dimensional quantized $(p_1, \ldots, p_k)$ th moments is then computed according to

$$E\left[\prod_{j=1}^{k} X_{Q_j}^{p_j}\right] = \frac{1}{i^{(p_1 + p_2 + \ldots + p_k)}} \frac{\partial^{(p_1 + p_2 + \ldots + p_k)}}{\partial u_1^{p_1} \partial u_2^{p_2} \ldots \partial u_k^{p_k}} [W_Q(u_1, u_2, \ldots, u_k)]_{u_1 = u_2 = \ldots = u_k = 0}$$

Now let's look at the effects of uniform quantizers on Gaussian random vectors. Let $X = [X_1, X_2, \ldots, X_k]^t$ be a zero mean k-dimensional Gaussian random vector with autocorrelation matrix R_X. Let $\hat{X} = \left[\hat{X}_1, \hat{X}_2 \ldots \hat{X}_k\right]^t$ be the output random vector of the

quantizer. Denote $E[X_iX_j]$, $E[\hat{X}_i\hat{X}_j]$, $E[\hat{X}_i^2]$ and $E[X_i^2]$ by σ_{ij}, $\hat{\sigma}_{ij}$, $\hat{\sigma}_i^2$ and σ_i^2 respectively. Then from (3-1) and (3-2):

$$\hat{\sigma}_j^2 = \sigma_j^2 + \Delta_j^2\left[\frac{1}{12} + \sum_{m=1}^{\infty} e^{-2m^2\pi^2\frac{\sigma_j^2}{\Delta_j^2}}\left(\frac{1}{m^2\pi^2} + \frac{4\sigma_j^2}{\Delta_j^2}\right)\cos(2\pi a_j m)\right] \quad (3\text{-}3)$$

and, for $i \neq j$,

$$\hat{\sigma}_{ij} = (1+\delta_{ij})\sigma_{ij} + \mu_{ij}, \quad (3\text{-}4)$$

where

$$\delta_{ij} = 2\left(\sum_{m_1=1}^{\infty} e^{-2m_1^2\pi^2\frac{\sigma_i^2}{\Delta_i^2}}\cos(2\pi a_i m_1) + \sum_{m_2=1}^{\infty} e^{-2m_2^2\pi^2\frac{\sigma_j^2}{\Delta_j^2}}\cos(2\pi a_j m_2)\right)$$

$$\mu_{ij} = \sum_{m_1,m_2=1}^{\infty} \frac{\Delta_i\Delta_j}{m_1m_2\pi^2} e^{-2\pi^2\left(\frac{m_1^2\sigma_i^2}{\Delta_i^2} + \frac{m_2^2\sigma_j^2}{\Delta_j^2}\right)}\left(\cos 2\pi a_i m_1 \cos 2\pi a_j m_2 \sinh\left(\frac{4\pi^2\sigma_{ij}m_1m_2}{\Delta_i\Delta_j}\right)\right.$$

$$\left. -\sin 2\pi a_i m_1 \sin 2\pi a_j m_2 \cosh\left(\frac{4\pi^2\sigma_{ij}m_1m_2}{\Delta_i\Delta_j}\right)\right)$$

Equations (3-3) and (3-4) specify the function C_Δ mentioned in the beginning of section 3.4. The function C_Δ shows the effects of uniform quantizers on the second moments of Gaussian random vectors.

3.4.2. Main Theorems

After the above preparations, in this section we will prove three main theorems for the proposed backward adaptive KLT transform coding algorithm.

In [18], Goyal, Zhuang, Vetterli and Chan proved the following theorem:

Theorem 3 Let $X = [X_1, \ldots, X_k]^t$, $X \sim N(0, \Sigma)$, where Σ is an unknown positive definite matrix. Let $\hat{X}$ be a scalar quantized version of X such that for $n \in Z$ either

(i) $X_i \in [n\Delta_i, (n+1)\Delta_i) \Rightarrow \hat{X}_i = \left(n + \frac{1}{2}\right)\Delta_i$; or

(ii) $X_i \in \left[\left(n - \frac{1}{2}\right)\Delta_i, \left(n + \frac{1}{2}\right)\Delta_i\right) \Rightarrow \hat{X}_i = n\Delta_i$.

Then for any set of positive, finite quantization step sizes $\Delta_1, \ldots, \Delta_k$, all moments of X can be recovered exactly from the first and second moments of $\hat{X}$.

Proof: For brevity, the proof below considers only case (i); case (ii) is similar. First note that since X is a Gaussian random variable, all its moments can be expressed in terms of its first and second order moments. Having already specified that X has zero mean, it is completely characterized by $\sigma_{ij} = E[X_i X_j]$, $1 \le i, j \le k$.

Simplifying expression from equations (3-3) and (3-4) gives

$$\hat{\sigma}_j^2 = \sigma_j^2 + \Delta_j^2 \left[\frac{1}{12} + \sum_{m=1}^{\infty} e^{-2m^2\pi^2\frac{\sigma_j^2}{\Delta_j^2}} \left(\frac{1}{m^2\pi^2} + \frac{4\sigma_j^2}{\Delta_j^2} \right) \right] \tag{3-5}$$

and, for $i \neq j$,

$$\hat{\sigma}_{ij} = (1 + \delta_{ij})\,\sigma_{ij} + \mu_{ij}, \tag{3-6}$$

where

$$\delta_{ij} = 2\left(\sum_{m_1 = 1}^{\infty} e^{-2m_1^2\pi^2\frac{\sigma_i^2}{\Delta_i^2}} + \sum_{m_2 = 1}^{\infty} e^{-2m_2^2\pi^2\frac{\sigma_j^2}{\Delta_j^2}} \right)$$

$$\mu_{ij} = \sum_{m_1, m_2 = 1}^{\infty} \frac{\Delta_i\Delta_j}{m_1 m_2 \pi^2} e^{-2\pi^2\left(\frac{m_1^2\sigma_i^2}{\Delta_i^2} + \frac{m_2^2\sigma_j^2}{\Delta_j^2}\right)} \sinh\left(\frac{4\pi^2\sigma_{ij}m_1 m_2}{\Delta_i\Delta_j}\right)$$

For any positive Δ_i, (3-5) describes a monotonic relationship between σ_i^2 and $\hat{\sigma}_i^2$, so each σ_i^2 can be determined from $\hat{\sigma}_i^2$. With σ_i^2 thusly determined, (3-6) describes a monotonic relationship between σ_{ij} and $\hat{\sigma}_{ij}$, so σ_{ij} can be similarly determined. □

Applying this theorem to recover the moments of the unquantized signal involves finds the roots of (3-5) and (3-6). This is clearly very complicated. There are two situations where we do not really have to find the roots of (3-5) and (3-6).

The first situation is when the quantization step sizes Δ_i, $i = 1, \ldots, k$ are small enough, (3-5) and (3-6) can be well approximated by $\hat{\sigma}_j^2 = \sigma_j^2 + \frac{1}{12}\Delta_j^2$ and $\hat{\sigma}_{ij} = \sigma_{ij}$ respectively.

The second situation is when, as we will show next, there exists an iteration scheme such that for any step size Δ below a certain threshold, the adapted transform exponentially converges to the true KLT transform. In practice, one iteration will result in transform close enough to the true KLT transform.

The next conjecture proposes that starting with an initial transform, without knowing the autocorrelation matrix of the source, our adaptive algorithm will update the transform in such a way that under certain conditions the adapted transform will exponentially converge to the true Karhunen-Loeve transform.

In the summer of 1996, after our ICIP'96 paper was published, we became very interested in the convergence properties of our proposed backward adaptive KLT algorithm. One simplification we made is that we assume the moments of the quantized random variables can be measured without noise. Based on this simplification, we then ask the following question: Will the adapted transform of our algorithm converge to the true Karhunen-Loeve transform?

The first intuition is that, because of the quantization effects, the autocorrelation matrix of the unquantized random vector is different from that of the quantized random vector, then the adopted transform will be definitely different from the true transform. However, our numerical simulations show that, surprisingly, the adapted transform converges to the true transform very fast, even for coarse uniform quantizer. We leave this as conjecture in this report, because, even though our simulation strongly support the convergence, the current proof is not satisfactory. Nonetheless, we give the current version in the following. We are currently working on providing a more rigorous proof of this conjecture.

Conjecture 4 Given the autocorrelation matrix of the source signal $R_x = T^t D T$, which has no repeated eigenvalues and an initial transform T_1, let $n = 0$,

step 1: $R_y = T_n R_x T_n^t$

step 2: $R_{\hat{y}} = C(R_y)$, where C is the function described by Cheded in [16], equations 43, 44, and 45.

step 3: $R_{\hat{x}} = T_n^t R_{\hat{y}} T_n$

step 4: $R_{\hat{x}} = T_{n+1}^t \Lambda_n T_{n+1}$

step 5: $n = n + 1$, go back to step 1

Then for $\forall R_x, \exists a > 0$, for $\forall \Delta < a$, $T_n \rightarrow T$ as $n \rightarrow \infty$.

Proof: First let's define:

definition 1: define Σ to be the set of all the diagonal matrix.

definition 2: for $\forall A$, $A = \left[a_{ij}\right]$, define $d(A, \Sigma) = \sum_{i \neq j} a_{ij}^2$

From the definitions, the following are true:

lemma 1: $A \in \Sigma$ if and only if $d(A, \Sigma) = 0$

lemma 2: $C(A) \in \Sigma$ if and only if $A \in \Sigma$

After the above preparations, we now begin to prove conjecture 4.

We have $R_{\hat{x}} = T_n^t C(R_y) T_n = T_n^t C\left(T_n R_x T_n^t\right) T_n = T_n^t C\left(T_n T^t D T T_n^t\right) T_n$

also we know that $R_{\hat{x}} = T_{n+1}^t \Lambda_n T_{n+1}$. Thus we have:

$$T_{n+1}^t \Lambda_n T_{n+1} = T_n^t Q\left(T_n T^t D T T_n^t\right) T_n$$

Rewrite the above equation and let $G_n = T_n T^t$ and $H_n = T_n T_{n+1}^t$, we have

$$C\left(G_n D G_n^t\right) = H_n \Lambda_n H_n^t$$

We can notice that:

$$G_n G_n^t = G_n^t G_n = I,\ H_n H_n^t = H_n^t H_n = I \text{ and } G_{n+1} = H_n^t G_n$$

Next, let $X_n = G_n D G_n^t$, then:

$$X_{n+1} = G_{n+1} D G_{n+1}^t = H_n^t G_n D G_n^t H_n = H_n^t X_n H_n$$

So far, we have:

$$C(X_n) = H_n \Lambda_n H_n^t$$

$$X_{n+1} = H_n^t X_n H_n$$

Next, let's define $Z_n = X_n - C(X_n)$

We have:

$$d(X_{n+1}, \Sigma) = d\left(H_n^t X_n H_n, \Sigma\right) = d\left(H_n^t (X_n - C(X_n) + C(X_n)) H_n\right)$$

Then:

$$d(X_{n+1}, \Sigma) = d\left(H_n^t (Z_n + C(X_n)) H_n, \Sigma\right)$$

Because we know that $H_n^t C(X_n) H_n = \Lambda_n$

$$d(X_{n+1}, \Sigma) = d\left(H_n^t Z_n H_n, \Sigma\right)$$

It is easy to verify that C has following properties:

(1) $Z_n \in \Sigma$ if and only if $X_n \in \Sigma$

(2) $d(Z_n, \Sigma) < \alpha d(X_n, \Sigma)$ and $\alpha \to 0$ when $\Delta \to 0$,

Next, we have:

$$d(X_{n+1}, \Sigma) = d\left(H_n^t Z_n H_n, \Sigma\right) = d(Z_n, \Sigma) + o(d(Z_n, \Sigma)) \qquad (3\text{-}7)$$

Readers interested in a detailed discussion of equation (3-7) are referred to Appendix B. Thus, for $d(Z_n, \Sigma)$ small enough, we have:

$$d(X_{n+1}, \Sigma) < 2d(Z_n, \Sigma) < 2\alpha d(X_n, \Sigma)$$

Thus for $2\alpha < 1$, we have $d(X_n, \Sigma) \to 0$ as $n \to \infty$. On the other hand, we already know that $X_n = G_n D G_n^t$ where $D \in \Sigma$ and $G_n G_n^t = G_n^t G_n = I$. Because the autocorrelation matrix R_x has distinct eigenvalues, it is only possible that $G_n \to I$. Thus we have $T_n \to T$ as $n \to \infty$ □

A detailed discussion in the two dimensional case of the above conjecture can be found in [25].

It is worthwhile to make a comment on equation 3-7 here. This assumption is supported by experimental results but we have not yet been able to prove it rigorously. Thus in this report, we use the word "conjecture" instead of "theorem". Currently, we are working on providing a rigorous proof for this conjecture.

Basically, our intuition about why the adapted transform converges is the following: First of all, the algorithm is stable (see theorem 5 below). Secondly, for a fine quantizer, $R_{\hat{x}} = R_x + \beta I$ is a good approximation of equations3-5 and 3-6 $R_{\hat{x}}$ and R_x have the same eigenvector matrices.

Finally, the next theorem shows the stability of the algorithm.

Theorem 5 [18] Let $X = [X_1, \ldots, X_k]^t$ be Gaussian random vector and have KLT T, i.e. T is the orthonormal matrix such that $TR_X T^t = \Lambda$, where $R_X = E[XX^t]$ and Λ is a diagonal matrix with non-increasing diagonal elements. Let $\hat{X} = T^t q(TX)$, where q is a scalar quantization function that introduces zero-mean quantization noise. Then, regardless of the quantization resolution, $TE[\hat{X}\hat{X}^t]T^t$ is a diagonal matrix, so the KLT of $\hat{X}$ differs from T by at most a permutation.

Proof: Let $Y = TX$ and $e = q(Y) - Y$. Note that since Y is Gaussian and has uncorrelated components, the components are independent. Since the quantization is scalar, we have furthermore that e_i is independent of e_j and Y_j for $i \neq j$. Then

$$TE[\hat{X}\hat{X}^t]T^t = TE[T^t q(Y)\, q(Y)^t T]T^t$$

Then $TE[\hat{X}\hat{X}^t]T^t = E[q(Y)\, q(Y)^t] = E[YY^t] + E[eY^t + Ye^t + ee^t]$

In the final expression, the first expectation gives Λ, while the three remaining expectations yield diagonal matrices because of the zero-mean and independence conditions. □

3.5. Practical Issues

3.5.1. Fast Algorithm to Estimate Autocorrelation Matrix

A key issue in the backward adaptive Karhunen-Loeve transform coding scheme is the measurement of the autocorrelation matrix. There is a fast algorithm which we can utilize. Here we give a brief review. Readers are referred to [24] for more details.

We begin with computation of the autocorrelation given by

$$r_i = \frac{1}{N}\sum_{k=0}^{N-1} d_k d_{k+i} \qquad i = 0, \ldots, \frac{n}{2} - 1$$

We are interested in problems in which the data blocklength N is much larger than the output blocklength $\frac{n}{2}$, so it would be wasteful to choose a Fourier transform with blocklength on the order of N. The sum can be broken into sections of length n in order to fit the problem to a smaller Fourier transform with blocklength n. Write

$$r_i = \sum_{l=0}^{L-1} r_i^{(l)}$$

where

$$r_i^{(l)} = \frac{1}{N}\sum_{k=0}^{\frac{n}{2}-1} d_{k+\frac{nl}{2}} d_{k+\frac{nl}{2}+i} \qquad i = 0, \ldots, \frac{n}{2} - 1 \qquad l = 0, \ldots, L-1$$

and $L\left(\frac{n}{2}\right) = N$. Thus the task is to compute the vector $r^{(l)}$ for $l = 0, \ldots, L-1$. This we do by computing appropriate cyclic convolutions using the FFT.

Let $d_i^{(l)} = d_{i+\frac{nl}{2}}$ for $i = 0, \ldots, \frac{n}{2} - 1$ and $d_i^{(l)} = 0$ for $i = \frac{n}{2}, \ldots, L-1$.

and

$$g_i^{(l)} = d_{i+\frac{nl}{2}} \qquad i = 0, \ldots, n-1$$

Then

$$r_i^{(l)} = \sum_{k=0}^{n-1} d_k g_{k+i} \qquad i = 0, \ldots, \frac{n}{2} - 1$$

Let $D^{(l)}$ and $G^{(l)}$ denote the Fourier transforms of $d^{(l)}$ and $g^{(l)}$. Then

$$S_k^{(l)} = D_k^{(l)} \otimes G_k^{(l)}$$

is the Fourier transform of the cyclic correlation

$$s_i^{(l)} = \sum_{k=0}^{n-1} d_k g_{((i+k))} \qquad i = 0, \ldots, n-1$$

and half of these values are the desired values

$$r_i^{(l)} = \frac{1}{N} s_i^{(l)} \qquad i = 0, \ldots, \frac{n}{2} - 1$$

Therefore we can first compute

$$S_k = \sum_{l=0}^{L-1} D_k^{(l)} \otimes G_k^{(l)} \qquad k = 0, \ldots, n-1$$

then compute its inverse Fourier transform s, and set

$$r_i = \frac{1}{N} s_i \qquad i = 0, \ldots, \frac{n}{2} - 1$$

To complete the development, we show how to eliminate some of the work. Instead of using a FFT to get $G^{(l)}$, use the formula

$$G_k^{(l)} = D_k^{(l)} + (-1)^k D_k^{(l+1)}$$

The computation then takes the form

$$S_k = \sum_{l=0}^{L-1} D_k^{(l)} \left[D_k^{(l)} + (-1)^k D_k^{(l+1)} \right]$$

An inverse Fourier transform completes the computation.

3.5.2. Fast Karhunen-Loeve Transform

In order to carry out a Karhunen-Loeve transform it is necessary to determine the covariance matrix of the data source. In practice, this will mean measuring the various variance products for the source, which is a task not lightly undertaken. Alternatively, a model for the source may be assumed, and the eigenvectors of the associated covariance matrix calculated. In either case, it is evident that the elements constituting the transform basis matrix are functions of the data properties, and so the matrix does not necessarily have any underlying structure and cannot, in general, be decomposed into sparse products to generate an algorithm with a reduced number of multiplications. It should be noted that, even when the eigenvectors can be expressed analytically, as for Markov processes, the equations lead to non-harmonic basis functions in general, and an underlying periodic structure is absent. A true "fast" algorithm for the KLT does not exist, therefore. However, several fast algorithms have been developed to approximate the KLT transform.

The first of these is that reported by Haralick et al [19]. Their treatment begins by generating the covariance matrix of the (two-dimensional) image in the manner described in?. The image is assumed to be stationary and isotropic, under which conditions the submatrix partitions of the main covariance matrix turn out to be (approximately) multiples of one another. Under this condition, each submatrix has the same eigenvectors, and the eigenvectors of the covariance matrix of the image as a whole are formed by the direct product of the submatrix eigenvectors and those of the matrix of eigenvectors. A submatrix is therefore generated such that the squared difference between the best multiple of its elements and those of a submatrix of the covariance matrix is minimized when averaged over all submatrix. Each submatrix is then replaced with the best multiple of the new submatrix. The direct product representation then allows decomposition of the transform operation into an equivalent fast process. Results using a 4 by 4 block size show that the true KLT is well approximated by the "fast" version, but also the difference between the true KLT and the slant transform, for example in terms of rms error is only a few tenth of 1%.

The second approach to the "fast" KLT is that of Jain [20] and is based upon the separation of a data vector $X = x_0 \ldots x_{n+1}$ into a boundary response x_b defined by x_0 and

x_{n+1}, and a partial sequence $X' - x_b$, where $X' = x_1 \ldots x_n$. Jain then shows that if X is a first order Markov sequence, the sequence $X' - x_b$ has, as its KLT, the sine transform. In this, or the more general two-dimensional, case the partial sequence (or field) is regarded as having been "pinned" at the boundary, giving rise to the so-called "pinned" KLT. Thus the "fast" KLT is in fact not so much a fast algorithm as such, but a decomposition of an image filed into sub-fields, for one of which the KLT is a (sinusoidal) fast transform.

An alternatively approach to the "fast" KLT has also been reported by Jain [21]. he points out that each member of the family of the sinusoidal transform is the KLT of a sequence having a particular J-matrix as its covariance matrix. Since the transform all have sinusoidal basis vector components, they are decomposable into fast algorithm structures. The success of the method rests upon the assumption that the measured image covariance matrix for the image or image sequence can be modelled as a simple function of a [J] matrix for which a fast transform exists.

3.5.3. Speed of Adaptation

The remaining block to be defined in the encoder of Fig. 3-1 is that in charge of determining the speed of adaptation. Our aim here is to dynamically determine at every iteration the number of past vector samples N that should be used in estimating the auto-correlation matrix.

The classes of error produced by the choice of memory can be separated into two classes:

(a) Non-significant data: if not enough memory is used we may be dealing with a non-significant (in a statistical sense) set of data and our estimation will necessarily be erroneous.

(b) Source with memory: if the source statistics (as determined by time averages over finite window) changes over time then an excess of memory will not permit sufficient adaptivity and will result in loss of performance.

One method to decide the speed of adaptation is to keep two sets of counters, one accumulating the long term statistics, the other accumulating the latest pattern of sample arrivals. We choose to use the short tern data to estimate the auto-correlation matrix only if the difference between short and long term data exceed a threshold. In this way, we try to detect the changes in statistics while avoiding always using a short term estimate, and thus risking having to deal with non-significant data.

3.6. Experimental Results

3.6.1. Universality and Convergence

Consider the following method for transform coding of a scalar source. The source samples are formed into vectors of size $k = 8$. These vectors are transformed by left multiplication by a orthonormal matrix T and then uniformly scalar quantized with step size Δ. Assume the quantized values will be coded with an adaptive entropy coder. At the decoder, and at the encoder for adaptation purposes, reconstruction is performed by left multiplication by T^t. After every block of N vectors, the transform T is updated to be the empirical KLT of the last m blocks of *quantized* data, where m is called *memory factor* and $m = \infty$ is used to indicate that all past data is used in the autocorrelation estimation.

This backward adaptive KLT coding scheme has been tested on zero-mean, unit variance AR(1) source. The results of the first experiment, given in Fig. 3-2, show that after a suitable amount of data has become available, close to optimal coding gain is achieved. The correlation coefficient of the source is $a = 0.9$, thus the optimal coding gain is 4.28. An initial transform of $T = I$, a block length of $N = 2$, a memory factor of $m = \infty$, and quantization step size $\Delta = 0.01, 0.5, 1$,and 2 were used. This experiment shows the relative importance of the number of samples available and the coarseness of the data. A system that uses unquantized data in adaptation (thus requiring side information) would have coding gain approximately as shown by the $\Delta = 0.01$ curve; the relatively small dif-

ference between this curve and the remaining curves shows that the dependence on the coarseness is quite small.

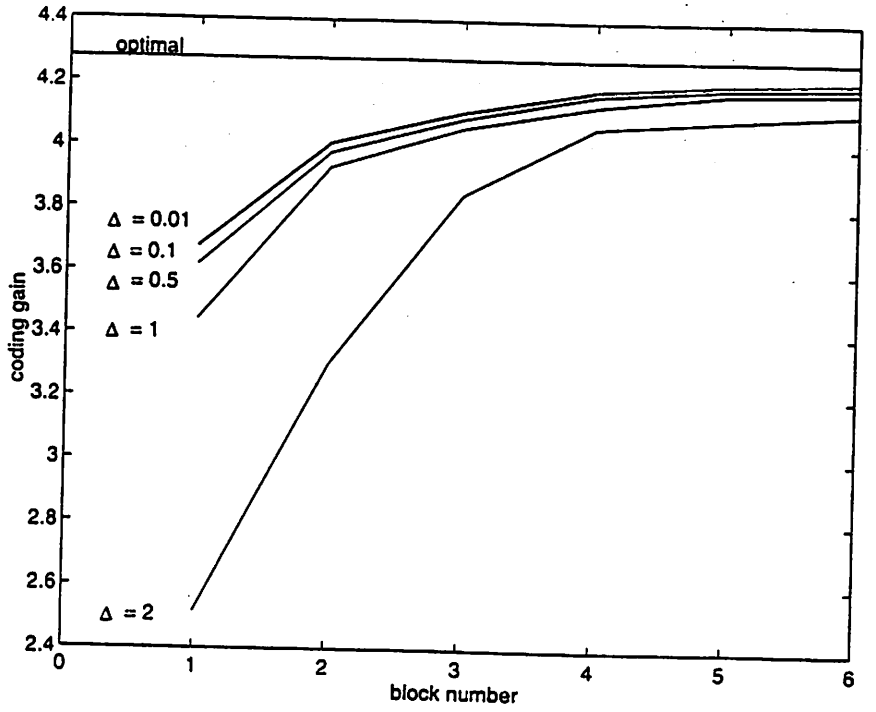

Figure 3-2. Coding gain of adaptive system compared to optimal coding gain for various coarseness levels

Conjecture 4 in the section 3.4.2 shows that if the autocorrelation matrix of the reconstructed vectors can be measured with infinite accuracy, then as the number of iterations goes to infinity, the adapted transform converges to the true Karhunen-Loeve transform when the quantizer step size Δ is small enough. In practice, however, the autocorrelation matrix can only be measured based on a finite number of reconstructed vector samples. In order to find out the behavior of the adapted transform when the measurement of the autocorrelation matrix is noisy, we set up the following simulation:

Let the source signal be an AR(1) signal with the correlation coefficient $a = 0.9$. We block the source into vectors of dimension $k = 4$. we set the quantization step size $\Delta = 1$ and set the adaptation interval to be 100, 200, 300 and 500 samples respectively. After each iteration, we compare the adapted transform with the true Karhunen-Loeve

transform and calculate the norm of the difference $\|T_n - T_{KLT}\|^2$. The results are shown in Fig. 3-3. As we can see, for a fixed quantization step size, the adapted transform in practice doesn't converge to the true Karhunen-Loeve transform. In theorem 5, we assumed the autocorrelation is measured based on an infinite number of samples and is therefore noiseless. This assumption is no longer valid in our simulations. Here, the autocorrelation is measured based on finite number of samples. Nonetheless, we can still see that even though the adapted transform doesn't converge to the true Karhunen-Loeve transform, it comes very close after one or two adaptations. Also, we can see that for a fixed quantization step size, as we increase the adaptation interval, the adapted transform becomes closer to the true KLT transform. It is reasonable to expect that the adapted transform will converge to the true KLT transform as the adaptation interval grows to infinity.

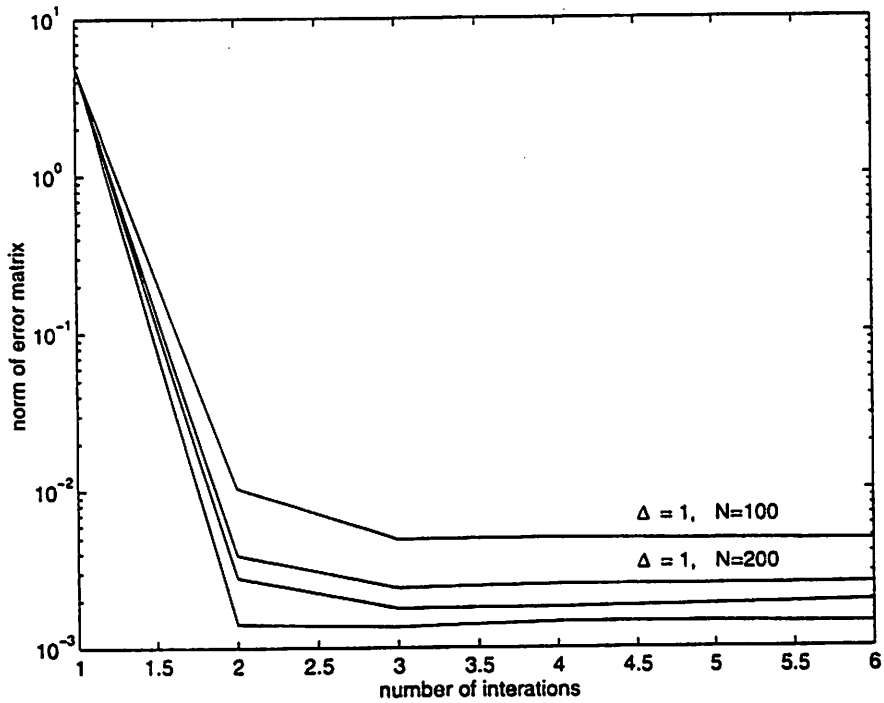

Figure 3-3. The adapted transform compared with the true KLT transform for fixed quantization step size and various adaptation intervals. The two curves at the bottom are $\Delta = 1, N = 300$ and $\Delta = 1, N = 500$, respectively.

Conjecture 4 shows that the smaller the quantization step size, the faster the adapted transform converges to the true Karhunen-Loeve transform. Fig. 3-4 shows the relation between quantization step size Δ and the norm of the error matrix $\|T_n - T_{KLT}\|^2$.

As we can notice in Fig. 3-4, for the first adaptation interval, the smaller the quantization step size, the smaller the norm of the error matrix. However, as the adaptation proceed, the norm of the error matrix when $\Delta = 3$ becomes smaller than when $\Delta = 0.1$. This means, the relation between the quantization step size and the norm of the error matrix is not a monotonous relation. The exact reason causing this non-monotonous relation is not clear and should be further investigated in the future.

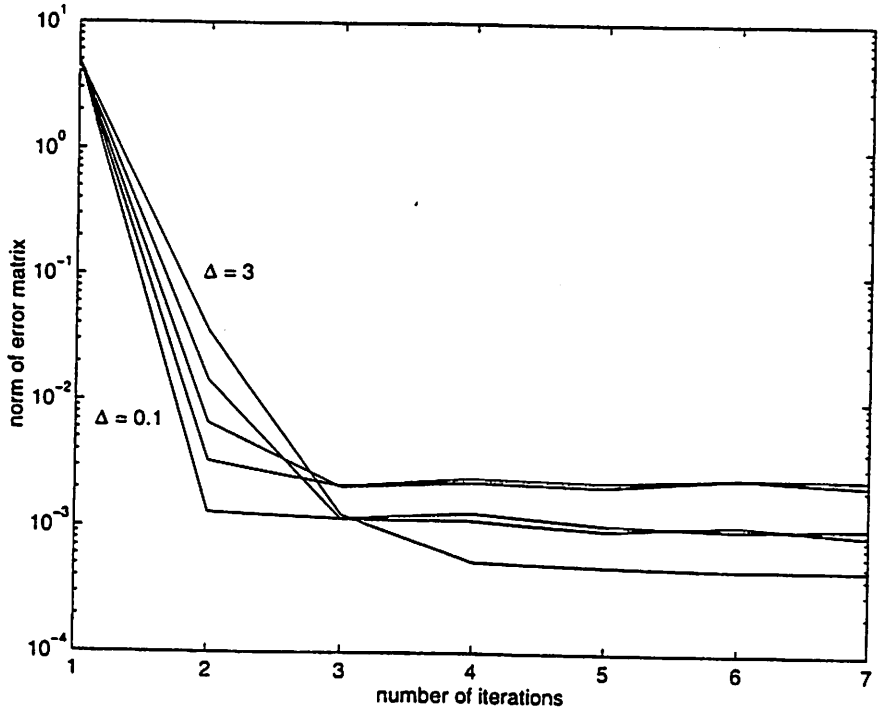

Figure 3-4. The adapted transform compared with the true KLT transform for fixed adaptation intervals ($N = 100$) and various quantization step sizes, $\Delta = 3$, $\Delta = 2$, $\Delta = 1.5$, $\Delta = 1$ and $\Delta = 0.1$. The three curves in between are, from left to right, $\Delta = 1$, $\Delta = 1.5$ and $\Delta = 2$.

In the above simulations, we have fixed the adaptation interval during the coding process. However, when coding stationary source. we can also exponentially increase the adaptation interval during the coding process. In the following simulation, we divide the source to blocks of size N, $2N$, $4N$, $8N$, ..., where $N = 100$. We then adapt the transform at the end of each block. The results are shown in Fig. 3-5. The quantization step size is $\Delta = 1$. The relation between the norm of the error matrix and the number of adaptations

is the solid line in the figure. In order to compare with the results obtained for fixed adaptation intervals, we include the results of Fig. 3-3 (dashed lines). As we can see, for a stationary source, when we increase the adaptation interval during th encoding process, the adapted transform goes to the true Karhunen-Loeve transform more quickly. Again, we can expect that when the adaptation interval becomes infinity, the adapted transform will converge to the true Karhunen-Loeve transform.

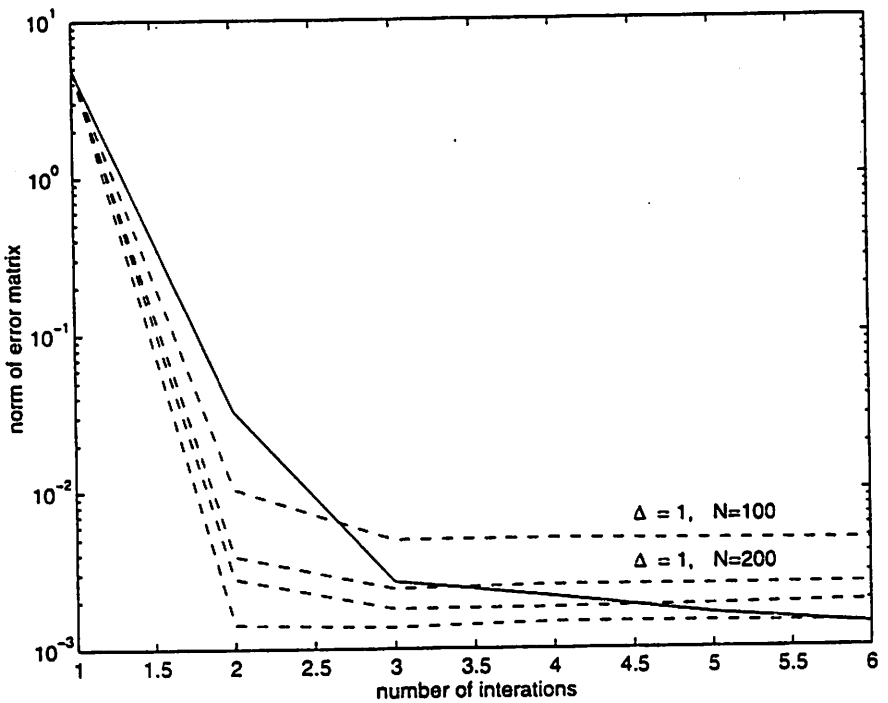

Figure 3-5. The adapted transform compared with the true KLT transform for exponentially growing adaptation interval.

3.6.2. Advantage of Adaptivity

A principal disadvantage of static coding algorithm is performance loss due to a mismatch between a source and the source assumed in the design. The second experiment demonstrates the advantage of using this adaptive method in the case of mismatch. The source has correlation coefficient $a = 0.99$ and is coded using an initial transform of $T = I$, a block of length of $N = 50$, and a memory factor of $m = 2$. In Fig.3-6, performance is compared to the performances of the static KLT coders designed for correlation coefficients of $a = -0.9, 0, 0.5, 0.9$, and the true value of 0.99. For the adaptive coder, the

rates and distortions given are averages over blocks 2 through 26. As the mismatch becomes larger, the performance gain from the adaptive system also becomes larger. The loss in the performance due to a properly matched static design is minimal.

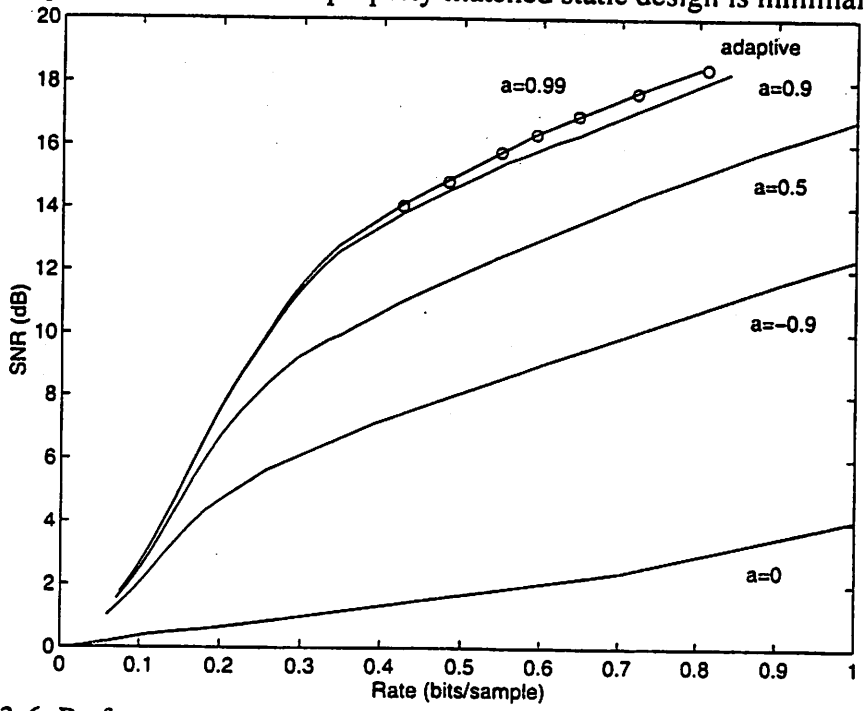

Figure 3-6. Performance comparison between adaptive system and fixed coders. Source has a correlation coefficient $a = 0.99$ and fixed coders are designed for various values of a, including the true value. o's mark performance of the adaptive system.

4 Conclusions and Future Work

This project report has two parts. In the first part, we have presented adaptive tree structured scalar quantization and adaptive TSVQ algorithms. We have shown that without knowing the distribution of the source, by splitting the encoder into model estimation, tree structured scalar quantizer design and update decision, an adaptive algorithm can designed.

In the second part we presented a backward adaptive Karhunen-Loeve transform coding algorithm. We first proved backward adaptive Karhunen-Loeve transform coding scheme exists in theory. Then we presented a practical backward adaptive algorithm which estimates the autocorrelation matrix of the causal past quantized random vectors. After analyzing the effects of uniform quantizer on the second moments of random variables, we proved that the algorithm is stable and for zero mean Gaussian random vector source and small quantization step size, the transform adapted by our proposed algorithm converges to the true Karhunen-Loeve transform. We have tested our adaptive algorithm in the coding of AR(1) source and have achieved promising results.

There are some issues that require further work.

First, in the adaptive TSVQ case, a future work would be to implement the adaptive scheme we have presented and apply it in image coding and compare the results with the performance of the static TSVQ.

Second, Karhunen-Loeve transform is the optimal transform under high resolution, real bit assignment and Gaussian signals assumptions. In this project we have proved the universality of our proposed algorithm and have tested it on synthetic data. Future work

should be to apply our adaptive algorithm in the coding of real image data and compare the results with the performance of the discrete cosine transform (DCT) coding.

In our proposed backward adaptive Karhunen-Loeve transform coding scheme, the size of adaptation interval is chosen quite arbitrarily. One of the future work should be to find out how to decide the size of the adaptation interval and adjust it during the coding process according to the change of source statistics.

Finally, in our backward adaptive Karhunen-Loeve transform coding scheme, we did not specify how to select the size of vectors. In fact, during the coding process, the size of vectors can be changed according to source statistics. When the source changes slowly, we can chose a large vector size and when the source changes quickly, we chose a small vector size. Thus one of the future work could be to find out how to chose the vector size and change the vector size according to local source statistics.

Appendix A

A. 1 Parameters Measurement Based on Quantized Samples in the Gaussian Case

First, it is easy to verify that $\sigma_1, \sigma_2, \mu_1, \mu_2$ have unique solutions in equations 3-1 (refer to [24]). So ρ is the only parameter need to be determined.

Let $p_{ij} = \int\limits_{\Omega_{ij}} \int f(x_1, x_2)\, dx_1 dx_2$, then

$$\frac{dp_{ij}}{d\rho} = A \int\limits_{\Omega_{ij}} \int \frac{1}{\left(1-\rho^2\right)^{\frac{5}{2}}} Q(x_1, x_2) \left[\left(1+\rho^2\right)x_1 x_2 - \rho x_1^2 - \rho x_2^2 + \rho - \rho^3\right] dx_1 dx_2$$

where

$$Q(x_1, x_2) = \exp\left\{-\frac{1}{2}\frac{1}{1-\rho^2}\left(x_1^2 - 2\rho x_1 x_2 + x_2^2\right)\right\}$$

It is easy to verify that for $\rho > 0$, $\exists \Omega_{ij}$ such that $\frac{dp_{ij}}{d\rho} < 0$ and for $\rho < 0$, $\exists \Omega_{ij}$ such that $\frac{dp_{ij}}{d\rho} > 0$. This implies that for ρ, there exists a quantizer whose p_{ij} 's uniquely determines ρ.

Thus, $\sigma_1, \sigma_2, \mu_1, \mu_2$ and ρ have unique solution in equations (3-1).

Appendix B

B. 1 Justification of Equation 3-7

Notations:
(1) denote the function described by Cheded by Q. That is, for a given matrix X, the function maps it to $Q(X)$.

(2) define $\|X\|^2 = \sum x_{ij}^2$

We are interested in the case when Δ, the quantization step size, is small. As we can verify, for small Δ, we can write $Q(X) = X + \frac{\Delta^2}{12} I + o\left(\Delta^t\right) C(X)$, where t is any positive integer and $C(X)$ is another function of X and $\|C(X)\| = 1$.

We already know that (refer to previous proof):

$Q(X_n) = H_n \Lambda_n H_n^T$ and

$X_{n+1} = H_n^T X_n H_n$

Thus we have:

$$X_n + \frac{\Delta^2}{12} I + o\left(\Delta^t\right) C(X_n) = H_n \Lambda_n H_n^T, \qquad (1)$$

On the other hand, we have

$$X_{n-1} + \frac{\Delta^2}{12} I + o\left(\Delta^t\right) C(X_{n-1}) = H_{n-1} \Lambda_{n-1} H_{n-1}^T$$

which means

$$H_{n-1}^T X_{n-1} H_{n-1} + \frac{\Delta^2}{12} I + o\left(\Delta^t\right) H_{n-1}^T C(X_{n-1}) H_{n-1} = \Lambda_{n-1} \qquad (2)$$

We know that $H_{n-1}^T X_{n-1} H_{n-1} = X_n$

So after compare equations (1) and (2) we have

$$H_n \Lambda_n H_n^T = \Lambda_{n-1} + o\left(\Delta^t\right)\left[C(X_n) - H_{n-1}^T C(X_{n-1}) H_{n-1}\right]$$

This leads to

$$H_n = I + o\left(\Delta^s\right) D(H_n) \qquad (3)$$

where s is a positive integer and
$\|D(H_n)\| = 1$.

Thus we have

$$X_{n+1} = \Lambda_n - \frac{\Delta^2}{12} I - o\left(\Delta^t\right)\left[\left(I + o\left(\Delta^s\right) D^T(H_n)\right) C(X_n) \left(I + o\left(\Delta^s\right) D(H_n)\right)\right]$$

Which means

$$X_{n+1} = \Lambda_n + Z_n + o\left(\Delta^{t+s}\right) E(H_n, X_n) \quad (4)$$

where $\|E(H_n, X_n)\| = 1$.

remember we have

$$Z_n = -\frac{\Delta^2}{12} I - o\left(\Delta^t\right) C(X_n) \qquad (5)$$

Compare (4) and (5) we have

$$d(X_{n+1}, \Sigma) = d(Z_n, \Sigma) + o(d(Z_n, \Sigma))$$

Bibliography

[1] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* KLP,1991.

[2] R. G. Gallager, "Variation on a theme by Huffman," *IEEE Trans. Infom. Theory*, vol. IT-24, pp. 668-674, 1978.

[3] P. Elias, "Universal codeword sets and representation of integers," *IEEE Trans. Infom. Theory*, vol. IT-21, no. 2, pp. 194-203, Mar, 1975.

[4] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 337-343, 1977.

[5] T. A. Welch, "A technique for high-performance data compression," *IEEE Comput.*, vol. 17, no. 6, pp. 8-19, June, 1984.

[6] J. J. Rissanen, "Universal coding, information, prediction, and estimation," *IEEE Trans. Infom. Theory,* vol. IT-30, no. 4, pp. 629-636, July, 1984.

[7] D. L. Neuhoff, R. M. Gray, and L. D. Davisson, "Fixed rate universal block source coding with a fidelity criterion," *IEEE Trans. Infom. Theory*, vol. IT-21, pp. 511-523, 1975.

[8] J. C. Kieffer, "A unified approach to weak universal source coding," *IEEE Trans. Infom. Theory*, vol. 24, pp. 674-682, 1978.

[9] D. S. Ornstein and P. Shields, "Universal almost sure data compression," *Annual Probab.*, vol. 18, no. 2, pp. 441-452, 1990.

[10] J. Ziv, "Coding for sources with unknown statistics: Part I. Probability of error," *IEEE Trans. Infom. Theory*, vol. IT-18, pp. 384-389, 1972.

[11] J. Ziv, "Coding for sources with unknown statistics: Part II. Distortion relative to a fidelity criterion," *IEEE Trans. Infom. Theory*, vol. IT-18, pp. 389-394, 1972.

[12] C. Chan and M. Vetterli, "Lossy compression of individual signals based on string matching and one pass codebook design," *Proceedings of ICASSP'95* (Detroit, MI) May 1995.

[13] A. Ortega and M. Vetterli, "Adaptive scalar quantization without side information," *Proceedings of ICIP 1994* (Austin, Tx), Nov, 1994.

[14] ITU-T Recommendation H.261, "Video codec for audiovisual services at $p \times 64$ kbit/s," Dec. 1990, March 1993 (revised).

[15] ISO/IEC 11172-2, "Information technology -- Coding of moving picture and associated audio for digital storage media at up to 1.5 Mbit/s: Part 2 Video," August 1993.

[16] L. Cheded and P. A. Payne, "The exact impact of amplitude quantization on multi-dimensional, high order moments estimation," Signal Processing, 39(1994) 293-315.

[17] L.Cheded, "Stochastic quantisation: Theory and application to moments recovery," Ph. D Thesis, University of Manchester (UMST), UK, 1988.

[18] V. Goyal, J. Zhuang, M. Vetterli and C. Chan, "Transform coding using adaptive bases and quantization," *Proceedings of ICIP 1996* (Lausanne, Switzerland), September, 1996.

[19] R. G. Harlick and N. Kattiyakulwanich, "A fast two-dimensional Karhunen-Loeve Transform," *SPIE Semin. Proc.* 66, 144-159.

[20] A. Jain, "Some new techniques in image processing," in "Image Science Mathematics" (C. O. Wilde and E. Barrett, eds.), pp.201-223, Western Period., North Hollywood, California.

[21] A. Jain, "A fast KLT for a class of random processes," *IEEE Trans. Commun.* COM-24, 1023-1029.

[22] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[23] M. Effros and P. A. Chou, "Weighted universal transform coding: universal image compression with the Karhunen-Loeve transform," *Proceedings of ICIP 1995* (Washington, D.C., USA), October, 1995. Vol. II, pp. 61-64.

[24] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*. Addison-Wesley, 1987.

[25] V. Goyal, J. Zhuang, M. Vetterli, "Universal transform coding using backward adaptation," DCC'97, submitted.