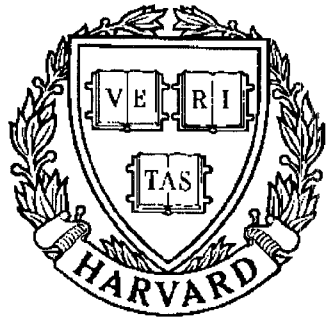
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



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

Research support for this report has been provided by in part by National Science Foundation grants NSFD MIP-86-57311 and NSFD CDR-85-00108 and in part by a grant from General Electric Company

Variable-Rate Finite-State Vector Quantization and Applications to Speech and Image Coding

by Y. Hussain and N. Farvardin

TR 91-61

Variable-Rate Finite-State Vector Quantization and Applications to Speech and Image Coding[†]

Y. Hussain and N. Farvardin [‡]
Electrical Engineering Department, and
Systems Research Center
University of Maryland
College Park, MD 20742

Abstract

A finite-state vector quantizer is a finite-state machine that can be viewed as a collection of memoryless full-searched vector quantizers, where each input vector is encoded using a vector quantizer associated with the current encoder state; the current state and selected codeword determine the next encoder state. It is generally assumed that the state codebooks are unstructured and have the same cardinality leading to a fixed-rate scheme [1]. In this paper, we present two *variable-rate* variations of the scheme in [1] with the possibility of using structured as well as unstructured state codebooks. In the first scheme, we let the state codebook sizes be different for different states, implying different rate distribution among the states. In the second scheme, in addition to this flexibility, we use pruned tree-structured vector quantizers as the state quantizers, i.e., we let each of the state quantizers be a variable-rate encoder. For encoding sampled speech data, both of these schemes perform significantly better than the fixed-rate scheme of [1]. The second scheme gives the best performance of all; performance improvements of up to 4.25 dB at the rate of 0.5 bits/sample are obtained over the scheme in [1].

We also consider the 2-D extension of the above mentioned schemes and describe two low bit rate image coding systems based on these schemes. The first system subtracts the mean from each input block and then encodes the mean-subtracted block by means of the 2-D versions of fixed-rate and variable-rate finite-state vector quantizer; the block-mean is separately encoded in an efficient manner by exploiting the high correlation present in the means of adjacent blocks. In the second system, a prediction is made on each pixel using a 5^{th}-order predictor and the residual is again encoded using the 2-D versions of the fixed-rate and variable-rate finite-state vector quantizer. At a bit rate of 0.3 bits per pixel, a peak signal-to-noise ratio in excess of 31 dB is achieved for encoding the 512×512 version of "Lena" using the schemes employing variable-rate finite-state vector quantizers.

[†]This work was supported in part by National Science Foundation grants NSFD MIP-86-57311 and NSFD CDR-85-00108 and in part by a grant from General Electric Company.

[‡]Currently on sabbatical leave at Ecole Nationale Superieure des Telecommunications, Paris, France.

1 Introduction

In the last decade an extensive amount of work has been done on vector quantization (VQ) as a means for data compression [2]-[6]. The main motivation behind the use of VQ was the result by Shannon [7] that VQ can attain performance close to the "best possible" in the rate-distortion theoretic sense in the limit when the block length goes to infinity. In a practical situation, however, we can only consider finite block lengths, and practical VQ systems fall short of achieving the best, while still performing much better than scalar quantizers. The main reason for the superior performance of VQ over scalar quantization is that VQ exploits the correlation between the components of the vector, while scalar quantization does not. Although the performance of the VQ at a given rate can be improved by increasing the vector dimension, the resulting complexity places a practical limit on how large a block size can be used. An alternative way to improve the performance of VQ (besides increasing the block size) for a given rate is to incorporate memory in the VQ structure [2]. One such technique is finite-state vector quantization (FSVQ) [1], [2], [8], which is basically a time varying VQ. There is a super-codebook that contains a very large number of codevectors and there is an internal state which accurately represents a small region (with a small sized subset of the super-codebook covering this region) that contains the source vector at any given time. Thus FSVQ achieves the efficiency of a large rate codebook at a relatively small rate.

In [1], the FSVQ was introduced and used to encode two sources with memory, namely the 1^{st}-order Gauss-Markov source and sampled speech waveform; performance improvements were observed as compared to the ordinary memoryless VQ (LBG-VQ). Then in [8], a technique based on adaptive stochastic automata theory was introduced that led to an improved design algorithm for FSVQ and an application of FSVQ was made to voice coding. Several schemes based on FSVQ have also been reported in the image coding literature. In [9] and [10], FSVQ was used to encode still images where the state was used to exploit the correlation in the spatial domain; over 50% saving in bit rate is achieved over LBG-VQ. In [11] and [12], FSVQ was used in coding image sequences where the state is defined to exploit temporal correlation; again a saving of over 50% is achieved over intraframe LBG-VQ.

In the FSVQ systems mentioned above, the state codebooks are all assumed to have the same cardinality and hence the same bit rate. This, in a loose sense, implies that all the states are treated with approximately equal degree of fidelity. Such a restriction limits the performance of the FSVQ which has the potential of doing better. In this paper, we have relaxed this assumption and have considered an FSVQ scheme with different codebook sizes

for different states while constraining the average bit rate. This modification has resulted in performance improvements (in some cases substantial) in terms of signal-to-noise ratio. We have also considered the possibility of using structured VQs such as the tree-searched VQ (TSVQ) [13] and optimally pruned unbalanced TSVQ (UTSVQ) [14] as the state quantizers. In fact, the scheme using UTSVQ as state quantizers and with the flexibility of having variable bit rate assignment among the states performs the best amongst all the schemes considered in this paper, as will be shown by the simulation results.

In order to facilitate the presentation and avoid using long names for different kinds of systems described in this paper, we will use appropriately defined acronyms. Furthermore, as an aid to remember the correspondence between different acronyms and their respective systems, we provide the notation tree of Figure 1. In this figure, FSVQ refers jointly to fixed-rate as well as variable-rate finite-state vector quantization; fixed-rate FSVQ is denoted by FR-FSVQ while variable-rate FSVQ is referred to by VR-FSVQ. Furthermore, FR-FSVQ with LBG-VQ as state quantizers is called FS-VQ and when TSVQs are used as state quantizers then the FR-FSVQ system is referred to as FS-TSVQ. Similarly, VR-FSVQ with LBG-VQ is called VFS-VQ and the one with TSVQ as state quantizers is named VFS-TSVQ; finally, we use VFS-UTSVQ to denote VR-FSVQ with UTSVQs as state quantizers.

This paper can be basically divided into two parts. In the first part, we describe the 1-dimensional (1-D) versions of various FR- and VR-FSVQs mentioned above and use them to encode sampled speech and a synthetic source described in Section 4. In the second part of the paper, we consider 2-dimensional (2-D) extensions of various FSVQ systems and their application to low bit rate image coding. It is shown by means of simulations that the image coding systems using VR-FSVQ give good performance results at low bit rates of 0.25-0.30 bits/pixel.

The rest of the paper is organized as follows. In Section 2, we provide the definition and design algorithm for FR-FSVQ with both structured and unstructured codebooks. The description and design algorithm for VR-FSVQ is provided in Section 3, while the simulation results of the various FSVQ based systems on 1-D sources like sampled speech waveform and a switched Gauss-Markov source are given in Section 4. Following that, we consider 2-D extensions of FR-FSVQ and VR-FSVQ in Section 5 and describe two low bit rate image coding systems based on 2-D FSVQ systems in Section 6. Section 7 provides the simulation results on images and finally a summary and conclusions are given in Section 8.

2 Fixed-Rate Finite-State Vector Quantization

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

2.1 Definition of FS-VQ

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

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

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

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

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

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

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

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

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

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

The encoder mapping is specified in terms of a distortion function that is used to measure the performance of the FS-VQ. The distortion measure $d : \mathbb{R}^L \times \hat{\mathcal{A}} \rightarrow [0, \infty)$ assigns a non-negative cost $d(\mathbf{x}, \hat{\mathbf{x}})$ to reproducing the input vector $\mathbf{x}$ as $\hat{\mathbf{x}}$. Then the encoder is specified by the minimum distortion rule [1]

$$\alpha(\mathbf{x}, k) = \arg\min_{u \in \mathcal{N}} d(\mathbf{x}, \beta(u, k)), \forall k \in \mathcal{S}. \tag{2}$$

An FS-VQ can be interpreted as a set of K LBG-VQs (one LBG-VQ associated with each state), each of codebook size N [2], [3]. The current input vector is vector-quantized using the LBG-VQ associated with the current state of the system; the current state and channel symbol determine the next state.

2.2 Design Algorithm of FS-VQ

Given a training sequence $\{\mathbf{x}_n,\ n = 0, 1, \ldots\}$ and a distortion measure, the design algorithm for an L-dimensional, K-state FS-VQ of rate $R = \log_2 N$, bits/vector consists of designing: (a) the state codebooks $\mathcal{C}_k$ for FS-VQ, each of size N, and (b) the next-state function $f(u, k)$, $u \in \mathcal{N}$, $k \in \mathcal{S}$. Following [1], the design algorithm can be described in four steps as described below.

1. Design an LBG-VQ [3] with K codevectors for the given training sequence. We refer to this VQ as the state-label VQ, $\mathcal{C} = \{\mathbf{c}(k), k \in \mathcal{S}\}$.

2. For each state k of the FS-VQ, design an initial reproduction codebook $\mathcal{C}_k = \{\beta(u, k), u \in \mathcal{N}\}$ using the LBG algorithm [3] on the subtraining sequence composed of all successors to vectors which are represented by k if quantized by the state-label VQ, i.e., the subsequence $\{\mathbf{x}_n : k = \arg\min_{s\in\mathcal{S}} d(\mathbf{x}_{n-1}, \mathbf{c}(s))\}$. Thus each codebook $\mathcal{C}_k$ is designed to be good for vectors which will occur next if the FS-VQ is currently in the ideal state k.

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

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

In most cases further improvements are possible by iterating steps (3) and (4). The algorithm described above does not necessarily converge and in fact it does not even guarantee

improved performance at each step of iteration. However, the FS-VQs designed with this algorithm exhibit substantial gain over LBG-VQs in both waveform coding and vocoding applications [1], [8].

2.3 FR-FSVQ Based on Tree-Structured Vector Quantizer

A finite-state tree-structured vector quantizer (FS-TSVQ) is specified in a manner very similar to FS-VQ. Associated with each state, we now have a TSVQ rather than an LBG-VQ. The encoding is accordingly done in a tree-structured manner. In particular, the encoder mapping $\alpha : \mathbb{R}^L \times \mathcal{S} \rightarrow \mathcal{N}$ for FS-TSVQ differs from that of the FS-VQ; rather than computing the index of the minimum-distortion codevector in the state codebook, we now encode the input vector using the state TSVQ and the output channel symbol is the index of the codeword resulting from the TSVQ encoding. The main advantage of this scheme is the complexity reduction in the encoding obtained due to the structured nature of the state codebooks without much loss in performance. The design algorithm for FS-TSVQ is very similar to that of FS-VQ given in Subsection 2.2. The only difference lies in the encoding procedure as mentioned above and in step (2) of the algorithm, where we now design a TSVQ for each state instead of an LBG-VQ. As in the case of FS-VQ, the design algorithm for FS-TSVQ does not necessarily converge and there is no guaranteed performance improvement at each step of iteration.

In the FS-VQ design algorithm presented by Foster *et al* [1], all the state codebooks are assumed to have the same cardinality and in order to design a rate R, bits/vector FS-VQ, the cardinality of each state codebook is assumed to be 2^R. In the next section, we drop this assumption and present a modified FS-VQ system (and also a modified FS-TSVQ system), in which the state codebook sizes are allowed to vary from state to state; this leads to performance gains (in some cases substantial). We also consider a second variation of the FS-VQ system in which UTSVQs obtained by optimal pruning of complete TSVQs [14] are used as state VQs and state rates are not constrained to be the same for all the states.

3 Variable-Rate Finite-State Vector Quantization

So far, the state VQs were assumed to have the same cardinality (and hence the same bit rate). Roughly speaking, this assumption implies that the source vectors are encoded with more-or-less the same degree of fidelity regardless of the state. This assumption may be unnecessarily restrictive in certain applications where some types of source vectors should be quantized more finely than some others (e.g., in speech coding silence periods can be

quantized quite coarsely). In what follows we relax this assumption and let the rates of state VQs vary from state to state subject to a constraint on the average encoding rate.

3.1 Definition of VR-FSVQ

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

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

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

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

Here the channel alphabet depends on the state ($\mathcal{N}(k)$ for state k) and, in general, is different for each state. Hence the system becomes a variable-rate system. Accordingly, the rate of the system is defined by $R = \sum_{k=1}^{K} P_k b_k$, bits/vector, where b_k is the average rate of the quantizer associated with state k and P_k is the probability of occurrence of state k. The operation of VR-FSVQ is the same as that of FR-FSVQ. It can again be interpreted as a set of K state quantizers, one associated with each state and the bit rate of the quantizer associated with state k is b_k. The current input vector is encoded using the quantizer associated with the current state of the system; the current state and the channel symbol determine the next state. In the sequel, we will refer to the variable-rate versions of FS-VQ and FS-TSVQ by VFS-VQ and VFS-TSVQ, respectively.

3.2 Design of VFS-VQ

A VFS-VQ differs from an FS-VQ due to the fact that now the bit rates associated with different state codebooks are not constrained to be same. As a consequence, state codebook sizes become an additional set of variables in the design stage of VFS-VQ. Once the codebook sizes associated with states are determined, the design algorithm for VFS-VQ can be described along the lines of the design algorithm for FS-VQ. The bit-rate assignment (codebook size determination) is done using the concept of optimal pruning of TSVQ [14] described next. Following that, we describe the bit assignment algorithm and the complete design algorithm for VFS-VQ.

3.2.1 Optimal Pruning of a TSVQ

Consider a complete binary TSVQ of rate l bits/vector. Corresponding to this TSVQ, there is a complete binary tree of depth l with 2^l leaves. Associated with each interior node (not including the root node) and leaf of the tree, there is a codevector (reproduction level), a

probability and a conditional expected distortion. By pruning off various branches of the tree, a variable-rate TSVQ or an unbalanced TSVQ (UTSVQ) is obtained. The codebook of the UTSVQ is the set of the codevectors associated with the leaves of the pruned tree. The quantizer's average rate is the sum, over all the leaves, of the leaf probability times the length from the root to the leaf. The quantizer's average distortion is the sum, over all the leaves, of the leaf probability times the conditional expected distortion associated with the leaf.

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

$$\hat{D}(R) = \min_{\mathcal{P} \preceq \mathcal{T}} \{\delta(\mathcal{P})/\ell(\mathcal{P}) \leq R\} \tag{4}$$

specifies the optimal trade-off between rate and distortion over all pruned subtrees of $\mathcal{T}$. A reinterpreted version of an algorithm developed in the context of classification and regression trees [15] is presented in [14] which traces out the convex-hull of the operational distortion-rate performance. The algorithm given in [14] is quite general and if δ is any monotone decreasing real-valued function defined on trees (i.e., if $\mathcal{P}_1 \preceq \mathcal{P}_2 \preceq \mathcal{T}$, then $\delta(\mathcal{P}_1) \geq \delta(\mathcal{P}_2)$) and ℓ is any monotone increasing real-valued function defined on trees, then the algorithm gives the optimal trade-off between ℓ and δ over all pruned subtrees of $\mathcal{T}$.

3.2.2 Bit Rate Assignment Algorithm

Suppose we are given K sets of collection of codebooks $\{\mathcal{C}_k^i, i = 1, 2, \ldots, M_k\}$, $k \in \mathcal{S}$, one set associated with each state. Let the rate and average distortion associated with $\mathcal{C}_k^i$ be given by $R_{k,i}$ and $D_{k,R_{k,i}}$, respectively. Also let $R_{k,1} < R_{k,2} < \ldots < R_{k,M_k}$, $\forall k \in \mathcal{S}$. Then, $D_{k,R_{k,1}} \geq D_{k,R_{k,2}} \geq \ldots \geq D_{k,R_{k,M_k}}$, $\forall k \in \mathcal{S}$. Given the set of codebooks $\{\mathcal{C}_k^i, i = 1, 2, \ldots, M_k\}, k \in \mathcal{S}$, we want to choose a codebook of bit rate b_k^* from each set as the state codebook of the VFS-VQ, i.e., determine the bit rate assignment map $(b_1^*, b_2^*, \ldots, b_k^*)$ (not necessarily integers) that minimizes the average distortion given by

$$D = \sum_{k=1}^{K} P_k D_{k,b_k}, \tag{5.a}$$

subject to

$$\sum_{k=1}^{K} P_k b_k \leq b_{avg}, \tag{5.b}$$

and

$$b_k \in \{R_{k,1}, R_{k,2}, \ldots, R_{k,M_k}\}. \tag{5.c}$$

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

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

$$\ell(\mathcal{P}) = \sum_{k=1}^{K} P_k R_{k,l_k}, \tag{6}$$

and the average distortion is given by

$$\delta(\mathcal{P}) = \sum_{k=1}^{K} P_k D_{k,R_{k,l_k}}. \tag{7}$$

The optimal pruning algorithm of [14], when applied to the tree $\mathcal{T}$ constructed above gives the optimal pruned subtree $\mathcal{P}^*$ and hence the bit rate assignment map $(b_1^*, b_2^*, \ldots, b_K^*)$ that minimizes $\delta(\mathcal{P})$ subject to $\ell(\mathcal{P}) \leq b_{avg}$ over all $\mathcal{P} \preceq \mathcal{T}$.

3.2.3 Design Algorithm for VFS-VQ

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

$$D = \sum_{k=1}^{K} P_k D_{k,b_k}, \tag{8}$$

subject to a constraint on the average rate described by

$$\sum_{k=1}^{K} P_k b_k \leq b_{avg}, \tag{9}$$

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

1. For the given training sequence, design the state-label VQ, $\mathcal{C} = \{\mathbf{c}(k), k \in \mathcal{S}\}$, using the LBG algorithm [3].

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

3. Find the optimum bit assignment map $(b_1^*, b_2^*, \ldots, b_K^*)$ using the algorithm described in Subsection 3.2.2. The state codebook used for state k will be $\mathcal{C}_k^{b_k^*} = \{\beta^{b_k^*}(u, k), u \in \{1, 2, \ldots, 2^{b_k^*}\}\}$ for VFS-VQ, where $\beta^{b_k^*}$ is the finite state decoder, associated with state k, with channel alphabet $\{1, 2, \ldots, 2^{b_k^*}\}$.

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

$$f(u, k) = \arg\min_{s \in \mathcal{S}} d(\beta^{b_k^*}(u, k), \mathbf{c}(s)), \; k \in \mathcal{S}, u \in \{1, 2, \ldots, 2^{b_k^*}\}. \tag{10}$$

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

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

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

As in the case of FS-VQ, the design algorithm does not necessarily converge and it does not even guarantee improvement at each step of iteration. However, the system obtained using this algorithm performs better than FS-VQ as will be shown by the results in Section 4.

3.3 VR-FSVQ Based on TSVQ

As in the fixed-rate case, we can describe a VR-FSVQ based on TSVQ (VFS-TSVQ) as a VFS-VQ in which each state LBG-VQ is replaced by a TSVQ. Again the advantage of VFS-TSVQ over VFS-VQ is the computational complexity reduction without much performance loss. The design algorithm for VFS-TSVQ is the same as described in Subsection 3.2.3 except that step (6) in the algorithm is replaced by the following step (6′):

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

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

3.3.1 VR-FSVQ Based on UTSVQs

In another variation of the FS-VQ scheme, we consider a system in which the state quantizers are optimally pruned UTSVQs obtained using the algorithm in [14]. We also have the flexibility of having different rates for different states. The main motivation behind using such a scheme was the superior performance of the optimally pruned UTSVQ over LBG-VQ along with the additional advantage of fast encoding due to the tree-searched method. In this scheme, even for a given state, the quantizer is a variable-rate encoder; for VFS-VQ (VFS-TSVQ) the rate varies between states but is fixed within each state. We refer to the new scheme as VFS-UTSVQ. The system is formally described in the same way as VFS-TSVQ. The design algorithm is also similar to that of VFS-TSVQ with step (2) modified in the following way:

2′. For each state k, we design a complete TSVQ of rate $\hat{b}_{max,k}$, where $\hat{b}_{max,k}$ is determined in the same way as $b_{max,k}$ is determined in the design of VFS-TSVQ. Then,

using the optimal pruning algorithm of [14] on each of the complete state TSVQs, we obtain K sets of collection of optimally pruned UTSVQs. The rate of the optimally pruned UTSVQs associated with state k varies from 1 bit/vector to $\hat{b}_{max,k}$ bits/vector and takes finitely many values which are not necessarily integers; the fact that rates are not constrained to be integers as in VFS-VQ (VFS-TSVQ) leads to an additional improvement factor. Then we apply the bit assignment algorithm on the collection of K sets of optimally pruned UTSVQs to obtain the optimal bit rate assignment map $(b_1^*, b_2^*, \ldots, b_K^*)$.

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

4 Simulation Results for 1-D Sources

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

4.1 Results on the Synthetic Source

For a simple stationary source such as the 1^{st}-order G-M source with correlation coefficient 0.9, we found experimentally that FR-FSVQ achieved a gain of about 1 dB over LBG-VQ in terms of SQNR. However, the performance of all VR-FSVQ schemes were found to be quite close to that of FR-FSVQ. For this simple source, it is observed that the performance of FS-VQ is within 1.0-1.6 dB of the rate-distortion function for the rates considered in this paper and therefore, we do not expect the variable-rate versions of FSVQ to achieve any significant gain over FS-VQ. Similar observations were made in a slightly different context by [14] for this G-M source. Therefore, for benchmarking purposes, we have considered a synthetic, more complex switched source in which at the beginning of each switching period a hidden mechanism chooses, according to a Markovian transition, between two 1^{st}-order

G-M sources with different variances; in [14], a similar switched source (however, with memoryless subsources) was considered. For the example given here, the switch transition probability matrix is chosen as

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

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

Table 1 shows the performance of FS-VQ. The LBG-VQ performance results are also included in the table for comparison. For $K = 8$, FS-VQ performs better than LBG-VQ by about 0.6-0.9 dB and the gain increases with K. Table 2 summarizes the performance of FS-TSVQ and TSVQ. As compared to FS-VQ, FS-TSVQ performs slightly worse due to the use of TSVQ in place of LBG-VQ.

Tables 3 and 4 summarize the performance of VFS-VQ and VFS-TSVQ, respectively. In these and subsequent tables, the numbers in parentheses indicate the achieved bit rate while b is the design bit rate. The performance improvements of the VFS-VQ (VFS-TSVQ) over the FS-VQ (FS-TSVQ) schemes are evident. Finally, Table 5 contains the performance results for VFS-UTSVQ, which performs the best among the five schemes. As compared to the second best scheme VFS-VQ, it achieves about 0.15-0.5 dB gain in terms of SQNR. Also, for comparison purposes, we have included the results of UTSVQ [14]. As a result of using finite-state vector quantization, a gain of over 1.5 dB in terms of SQNR is achieved over UTSVQ.

4.2 Results on Sampled Speech

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

The performance of FS-VQ is summarized in Table 6. Also for comparison, we include the performance results of LBG-VQ. As observed in [1], the FS-VQ scheme outperforms the LBG-VQ by over 2 dB and the gain increases with the number of states. Table 7 summarizes the performance of FS-TSVQ. In this case, as compared to FS-VQ, a slight degradation of performance can be observed. In general, however, the trend is similar to that of FS-VQ. In most cases, FS-TSVQ yields a SQNR within 1.0 dB of FS-VQ.

Table 8 summarizes the performance of VFS-VQ for different values of b and K. Comparison of Tables 6 and 8 indicates that at the same bit rate, VFS-VQ outperforms FS-VQ,

in general, by about 2.5 dB. Note that the discrepancy between the desired rate and the achieved rate decreases with the increase in the number of states; when the number of states is relatively small, the number of achievable points on the convex-hull given by the pruning algorithm in [14] is small.

The performance of VFS-TSVQ is illustrated in Table 9. It should be noted that the difference between the performance of VFS-TSVQ and VFS-VQ is smaller than the difference between FS-TSVQ and FS-VQ. The reason resides in the limitation on the size of the largest LBG-VQ (2048 codevectors in codebook) needed for VFS-VQs; this limitation is less severe for TSVQs.

Finally, we include the performance results of VFS-UTSVQ in Table 10. This system gives the best performance results among all the schemes considered in this paper. For all values of K considered here, VFS-UTSVQ outperforms FS-VQ by at least 3 dB and by as much as 4.25 dB for $b = 0.5$ bits/sample and higher. In order to visually compare the performance of the different systems, their rate-distortion performance on the training sequence are illustrated in Figure 3, clearly demonstrating the superior performance of the variable-rate based schemes.

The performance of various FSVQ schemes on an out-of-training test sequence is summarized in Tables 11-15. The test sequence was 67 seconds of speech uttered by a male speaker and sampled at 8 KHz. These results are also depicted in Figure 4. Study of these tables shows that even for out-of-training data, variable-rate FSVQs outperform their fixed-rate counterparts. It is important to note that in the variable-rate schemes, while the SQNR degrades for the out-of-training sequence, the actual bit rate is also lower than the design rate b. Comparison at the same bit rate (see Fig. 4) shows a gain of as much as 3 dB for VFS-UTSVQ and 2.4 dB for VFS-VQ and VFS-TSVQ over the fixed-rate FSVQs. For the variable-rate schemes, a simple feedback mechanism can be used to sense the instantaneous output rate and adjust the bit rate accordingly by adjusting the encoder structure slightly; a feature that does not exist in a fixed-rate system. More interestingly, for the VFS-UTSVQ system, the bit rate adjustment can be achieved simply by adding/pruning branches of state codebooks. This is an additional important advantage over other variable-rate FSVQ systems (besides giving better performance at the same rate).

5 Extension to 2-D and Image Coding Applications

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

extensions of VR-FSVQ.

5.1 Extension of FR-FSVQ to 2-D

In the case of a 2-D source such as an image, each input vector is typically a 2-D block (matrix of size $l \times l$) and unlike the 1-D case, it has more than one adjacent preceding neighbor. For efficient encoding of an input vector $\mathbf{x}_n$, it is essential to exploit the correlation with the adjacent vectors in the north ($\mathbf{x}_{n-L}$) and west ($\mathbf{x}_{n-1}$) directions, where L is the number of blocks in an image row (see Figure 5). In an FSVQ system, this is done by appropriately defining the state variable. To define the state variable, we associate with each input vector $\mathbf{x}_n$ an index $v_n \in \{1, 2, \ldots, K\}$ [2] and say that $\mathbf{x}_n$ is in "substate" v_n iff v_n is the index associated with $\mathbf{x}_n$. We then define the state $\mathbf{s}_n$ associated with $\mathbf{x}_n$ as a two-component vector [3] $\mathbf{s}_n = (v_{n-1}, v_{n-L})$, where v_{n-1} and v_{n-L} are, respectively, the substates associated with the west ($\mathbf{x}_{n-1}$) and north ($\mathbf{x}_{n-L}$) neighbors of $\mathbf{x}_n$. Note that $\mathbf{s}_n$ can be equivalently described by an index $k = K(v_{n-1} - 1) + v_{n-L}$ with $k \in \{1, 2, \ldots, K^2\}$.

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

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

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

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

Here, $\mathcal{N} \triangleq \{1, 2, \ldots, N\}$ is the channel alphabet of size N and $\hat{\mathcal{A}}$ is the reproduction space. This definition is a 2-D extension of the 1-D FS-VQ (FS-TSVQ) given in [1], [16]. Note that the next-state map $\mathbf{f}$ has two components f_1 and f_2 determining the substates associated with the west and north neighbors, respectively. The collection $\mathcal{C}_k \triangleq \{\beta(u, k), u \in \mathcal{N}\}$ is the codebook associated with state k; obviously, $\hat{\mathcal{A}} = \bigcup_{k=1}^{K^2} \mathcal{C}_k$.

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

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

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

according to:

$$
\begin{aligned}
u_n &= \alpha(\mathbf{x}_n, \mathbf{s}_n), && \text{(11.a)}\\
\hat{\mathbf{x}}_n &= \beta(u_n, \mathbf{s}_n), && \text{(11.b)}\\
\mathbf{s}_{n+1} &= (f_1(u_n, \mathbf{s}_n), f_2(u_{n+1-L}, \mathbf{s}_{n+1-L})), \\
&\quad n = m(L+1), m(L+2), \ldots, \; m = 1, 2, \ldots. && \text{(11.c)}
\end{aligned}
$$

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

$$\alpha(\mathbf{x}, k) = \arg\min_{u \in \mathcal{N}} \; d(\mathbf{x}, \beta(u, k)), \forall k \in \{1, 2, \ldots, K^2\}, \tag{12}$$

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

5.2 Design Algorithm for 2-D FR-FSVQ

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

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

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

The design algorithm is an extension of the 1-D algorithm given in Section 2.2 and can be described in the following four steps.

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

$$k_1 = \arg \min_{v \in \{1,2,\ldots,K\}} d(\mathbf{x}_n, \mathbf{c}(v)). \tag{13}$$

2. For each state index $k \in \{1, 2, \ldots, K^2\}$ of the FS-VQ (FS-TSVQ), design an initial reproduction codebook $\mathcal{C}_k = \{\beta(u, k), u \in \mathcal{N}\}$ using the LBG algorithm [3] on the subtraining sequence $\{\mathbf{x}_n : k_1 = \arg\min_{v \in \{1,2,\ldots,K\}} d(\mathbf{x}_{n-1}, \mathbf{c}(v))$ and $k_2 = \arg\min_{v \in \{1,2,\ldots,K\}} d(\mathbf{x}_{n-L}, \mathbf{c}(v))\}$, where (k_1, k_2) is the pair associated with k (i.e., $k = K(k_1 - 1) + k_2$). Thus each codebook $\mathcal{C}_k$ is designed to be good for vectors whose west and north neighbors are associated with substates k_1 and k_2, respectively.

3. In order to enable the decoder to track the state sequence, we let the state sequence depend on the reproduction vector of the input vector rather than on the input vector itself. Thus given the substate-labels $\mathbf{c}(v)$ and the decoder β designed in step (2), we define the mappings f_1 and f_2 and hence the next-state function $\mathbf{f} = (f_1, f_2)$ by [4]

$$f_1(u,k) = f_2(u,k) = \arg \min_{v \in \{1,2...,K\}} d(\beta(u,k), \mathbf{c}(v)),\ k \in \{1,2,\ldots,K^2\}, u \in \mathcal{N}. \quad (14)$$

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

5.3 Description of 2-D VR-FSVQ

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

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

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

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

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

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

$$D = \sum_{k=1}^{K^2} P_k D_{k,b_k}, \quad (15)$$

subject to a constraint on the average rate described by

$$\sum_{k=1}^{K^2} P_k b_k \leq b_{avg}, \quad (16)$$

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

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

While the 1-D versions of FSVQ have been successfully applied to encoding of sampled speech [1], [16], direct application of 2-D versions of FSVQ on image data presents certain design problems. In what follows, we describe these problems and propose two methods to tackle them.

6 System Description

When the LBG algorithm [3] is used to design a small-sized LBG-VQ using a training sequence of images, the majority of the codevectors in the LBG-VQ correspond to the constant background vectors of different grey-levels and other feature vectors such as the edge vectors are either averaged out or masked by the background vectors [17]. Therefore in step (1) of the design algorithm for 2-D FR-FSVQ, the majority of the substate-labels (i.e., the codevectors in the substate-label VQ) will correspond to the constant background vectors and the states will then correspond to the various combinations of constant background.

To account for the various grey-levels of background requires a very large number of states. To alleviate this problem and more importantly to avoid the masking of edge vectors by background vectors, we have considered two methods. In the first method we subtract off the block-mean from each input vector and encode the block-mean and the residual separately [17]. Since the block-means of adjacent vectors are highly correlated, they can be encoded with a small number of bits; the residual is encoded using 2-D FS-VQ and VFS-UTSVQ. We will use ME-FS-VQ and ME-VFS-UTSVQ to denote, respectively, the systems in which FS-VQ and VFS-UTSVQ are used to encode the residual signal. In the second method we use a predictor [18] to make a prediction of each pixel of the input vector based on the knowledge of the already encoded neighboring vectors and then encode the residual using the above mentioned FSVQs. We will refer to the system using FS-VQ by PR-FS-VQ, while PR-VFS-UTSVQ will be used for the system with VFS-UTSVQ.

As a result of block-mean subtraction or prediction, all constant background vectors will result in residual vectors close to the zero grey-level and they can be classified by using fewer number of states. The rest of the states can be devoted to other feature vectors. As compared to other vectors, the residual vectors corresponding to the near zero grey-level background can be encoded using relatively smaller number of bits while achieving

comparable distortion making the residual vector sequence more amenable to variable-length coding. Details of the two systems are provided in the following subsections.

6.1 Encoding of Block-Mean

Typical real-world images exhibit high pixel-to-pixel correlation. For a vector of size 4×4 (the vector size considered throughout this paper for images), the block-mean of an input vector is also highly correlated with the block-mean of the adjacent vectors. As a consequence, the block-mean can, in turn, be efficiently encoded by a VQ. However, due to the complexity associated with the VQs, we cannot design a very large size codebook. For instance, in order to achieve a rate of 1 bit/sample (for blocks of size 4×4 of block-means), the number of codewords needed is 2^{16}, which is prohibitively large. To alleviate this problem, the block-mean is encoded in two steps: First, 4×4 blocks of block-mean are vector quantized using an LBG-VQ of small size and then the difference is encoded using an entropy-constrained block transform coding system. The entropy-constrained block transform coding system is similar to the system described in [19]. The only difference lies in the computation of the variances associated with the transform coefficients. In the present system depending on the codevector used to encode the block-mean vector we have a set of variances of the transform coefficients estimated by using a training sequence different from the test sequence.

6.2 Description of the Predictor

We used the 5^{th}-order predictor proposed in [18] to make the block prediction. Each pixel Y in an input vector (Figure 6) is predicted using only five other pixels. The predicted value of Y is given by $\hat{Y} = \mathbf{aX}$, where $\mathbf{X} = [X_1, X_2, X_3, X_4, X_5]$. X_1 and X_2 are the two closest pixels in the same row of the vector in the west, X_3 is the lower right pixel of the upper left diagonal subblock and X_4 and X_5 are the two closest pixels in the same column of the north neighbor vector. The vector of linear prediction coefficients $\mathbf{a}$ is chosen to minimize the mean squared-error between Y and $\hat{Y}$ according to $\mathbf{a} = \mathbf{R}^{-1}\mathbf{d}$ [20]. Here $\mathbf{R} = E(\mathbf{X}^{\mathrm{T}}\mathbf{X})$ and $\mathbf{d} = E(\mathbf{X}Y)$. Spatial stationarity is assumed for computing the correlation matrix. We use the actual values of $\mathbf{X}$ to compute $\mathbf{a}$ but once $\mathbf{a}$ is computed $\hat{Y}$ is always estimated based on the reconstructed version of $\mathbf{X}$ as in any DPCM system.

6.3 Encoding of the Residual Signal

We have used FS-VQ and VFS-UTSVQ for encoding the residual signal $\{\tilde{\mathbf{x}}_n\}$ obtained after block-mean subtraction or using 5^{th}-order prediction. When an input vector is encoded by means of an FSVQ, both linear and nonlinear dependence of the input vector on its neighbors can be exploited by the FSVQ. The use of block-mean subtraction or a linear predictor removes linear dependence of the input vector on its neighbors to a certain extent; however the residual vector still retains the nonlinear dependence and even some linear dependence (due to the use of a coarse predictor to keep overhead information low) on its neighbors that can be efficiently exploited by the use of FSVQ on the residual sequence.

The FS-VQ is designed for $\{\tilde{\mathbf{x}}_n\}$ by using the algorithm described in Subsection 5.2 and the encoding is done according to equations (11.a), (11.b) and (11.c). The simulation results are presented in the next section.

As for the VFS-UTSVQ, we have considered two kinds of UTSVQs. In the first case, we design a complete TSVQ and then use the optimal pruning algorithm of [14] to obtain the UTSVQ of the desired rate. In the second case, we use the greedy algorithm of [21] to grow an unbalanced tree (instead of a complete TSVQ of the same rate) and then use the pruning algorithm of [14] to design the UTSVQ. For the reasons described in [21], the second type of UTSVQ, in general, gives better performance. We will refer to the ME-VFS-UTSVQ system based on the first and the second kind of UTSVQs by ME-VFS-UTSVQ1 and ME-VFS-UTSVQ2, respectively. Corresponding names for the prediction based systems are PR-VFS-UTSVQ1 and PR-VFS-UTSVQ2, respectively. The simulation results for various systems are given in the next section.

7 Simulation Results on Images

The performance results are reported in terms of the peak signal-to-noise ratio (PSNR) in dB and the overall average bit rate per pixel (including the overhead) denoted by b. The design values selected were $l = 4$ and $K = 4$. This corresponds to a vector dimension of 16 and the number of states is 16. The database used for training various kinds of FSVQ systems consisted of 19 monochrome images of size 480×512 and the test image chosen was the 512×512 version of "Lena" not included in the training sequence.

7.1 Performance Results of ME-FS-VQ and ME-VFS-UTSVQ

The VQs designed using the residual training sequence obtained by subtracting off the block-mean from each input vector will have zero-mean codevectors. As a consequence,

for the squared-error distortion measure, it is easy to check that the overall distortion of ME-FS-VQ and ME-VFS-UTSVQ systems is equal to the sum of distortions that result from encoding the block-mean and the residual signal. Therefore, the problem of encoding the block-mean and that of encoding the residual can be treated separately.

The block-mean was encoded using the system that consisted of a VQ followed by entropy-constrained block transform coding described briefly in the Subsection 6.1. The block-mean corresponding to the adjacent vectors are grouped together in block sizes of 4×4 and vector quantized with a VQ consisting of 8 codevectors. The remaining bits were allocated to entropy-constrained block transform coding. We allocated $\frac{1.0}{16}$ to $\frac{2.0}{16}$ bits/pixel to encode the block-mean. For smaller values of b, the bit rate allocated for block-mean encoding was kept close to $\frac{1.0}{16}$ bits/pixel since any further increase in the bit rate for block-mean encoder resulted in a relatively small decrease in the overall distortion as the major contribution to the overall distortion came from the encoding of the residual. On the other hand, for higher values of b, distortion arising out of the encoding of the residual signal is reduced substantially and the distortion resulting from block-mean encoding starts playing an equally important role in contributing to the overall distortion. As a consequence, the bit rate associated with the block-mean encoder is increased to as much as $\frac{2.0}{16}$ for $b = 0.38$ bits/pixel; any further increase in the bit rate for block-mean encoding results in a very small decrease in overall distortion.

The residual sequence was encoded using FS-VQ and VFS-UTSVQ. Table 16 illustrates the results for various systems at different bit rates. The term in the parentheses is the value of the bit rate per pixel actually achieved by the system, while the desired value is given by b. We have also provided the results for the case when the residual sequence is encoded using an LBG-VQ (ME-VQ) for comparison purposes. Table 16 shows that on the average, ME-FS-VQ performs better than ME-VQ by about 0.8 dB at all bit rates shown in the table and ME-VFS-UTSVQ1 improves the performance by about 1 dB over ME-FS-VQ. The best performance is attained by ME-VFS-UTSVQ2; it outperforms ME-VQ by about 3 dB at $b = 0.31$ bits/pixel achieving a PSNR of 31.66 dB. The reconstructed image at $b = 0.31$ bits/pixel using ME-VFS-UTSVQ2 is shown in Figure 7.b; Figure 7.a shows the original image.

7.2 Performance Results of PR-FS-VQ and PR-VFS-UTSVQ

The PR-VFS-UTSVQ1 system is basically a more general version of the scheme described in [18]. In both these systems, an error sequence is formed by using a 5^{th}-order predictor. In [18], the error (residual) sequence is encoded using a UTSVQ obtained by optimally

pruning a complete TSVQ [14]; this system can be looked upon as PR-VFS-UTSVQ1 with just one state.

We have considered the encoding of the residual using LBG-VQ, FS-VQ and the two types of VFS-UTSVQ. The results are summarized in Table 17. We have also simulated and included the results of the system described in [18] (denoted by RDG) for comparison. The bit rate b here also includes the overhead information which consists of transmitting the actual value of each input sample in the first row vector and the first column vector of the image (about $\frac{0.125}{16}$ bits/pixel). Table 17 shows that use of PR-FS-VQ as opposed to PR-VQ leads to an increase of 0.6 dB in PSNR. For $b = 0.20$ and 0.26 bits/pixel, PR-VFS-UTSVQ1 outperforms PR-FS-VQ by over 1.5 dB. However, this gain is reduced at higher rates. This reduction in gain can be attributed to the limitation posed by the size of the training sequence used. As opposed to the ME-VFS-UTSVQ systems, PR-VFS-UTSVQ2 shows an insignificant improvement over PR-VFS-UTSVQ1 system. Both PR-VFS-UTSVQ1 and PR-VFS-UTSVQ2 systems perform better than the system described in [18] at the rates shown in the table and the improvement is most noticeable at $b = 0.20$ and 0.26 bits/pixel, both visually and in terms of PSNR. Figure 7.c shows the reconstructed image for PR-VFS-UTSVQ2 at $b = 0.32$ bits/pixel. The performance of PR-VFS-UTSVQ systems saturates above $b = 0.32$ bits/pixel due to the training sequence size constraint and therefore we have not included the results for higher rates in the table.

8 Summary and Conclusions

In this paper, we have considered several variable-rate variations of the 1-D FS-VQ scheme described in [1]. We have also considered the extension of these schemes to 2-D. Design algorithm for various schemes are obtained by appropriately modifying the algorithm given in [1]. 1-D versions of FSVQ systems are used for encoding sampled speech, while 2-D versions are used to develop low bit rate image coding systems. It can be concluded from our results that, in general, the variable-rate versions of FSVQ lead to performance improvements which in certain cases are quite substantial. The best performance was achieved by the systems based on VR-FSVQ using UTSVQs as the state quantizers for both speech and image coding. For image coding, out of all the schemes considered, PR-VFS-UTSVQs have the potential of doing the best.

Although the extension of fixed-rate FSVQ to VR-FSVQ leads to performance improvements, the potential problems associated with any variable-rate coding system like buffer overflow/underflow and channel error propagation will come into play. One possible method

to deal with these problems is to use the idea of the recently developed structured vector quantizer [22], [23] to convert the variable-rate FSVQ systems into a fixed-rate system without introducing a significant performance loss. By incorporating some delay into the system, the codebook search and encoding algorithm of [22] can be used to encode several vectors (say n) at a time such that the total number of bits used for the n vectors is fixed, but the number of bits used for each vector can be allowed to be variable. This modification will render the system, a fixed-rate encoder while at the same time offers (to some extent) the advantages of the variable-rate systems. As a consequence, the effect of any channel error will remain confined to only n vectors; similarly the buffer overflow/underflow problem is eliminated since the encoder is a fixed-rate system now. Work on the development of this modification is currently underway.

References

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

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

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

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

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

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

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

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

[9] R. Aravind and A. Gersho, "Low-Rate Image Coding with Finite-State Vector Quantization," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Processing*, pp. 137-140, April 1986.

[10] T. Kim, "New Finite State Vector Quantizers for Images," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Processing*, pp. 1180-1183, April 1988.

[11] R. L. Baker and H. H. Shen, "A Finite-State Vector Quantizer for Low Rate Image Sequence Coding," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Processing*, pp. 760-763, May 1987.

[12] H. H. Shen, and R. L. Baker, "A Finite State/Frame Difference Interpolative Vector Quantizer for Low Rate Image Sequence Coding," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Processing*, pp. 1188-1191, April 1988.

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

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

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

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

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

[18] E. A. Riskin, E. M. Daly and R. M. Gray, "Pruned Tree-Structured Vector Quantization in Image Coding," *Proc. IEEE Int. Conf. Acoust., Speech and Signal Processing*, Glasgow, Scotland, pp. 1735-1738, May 1989.

[19] Y. Hussain and N. Farvardin, "Adaptive Block Transform Coding of Speech Based on LPC Vector Quantization," *IEEE Trans. Acoust. Speech Signal Process*, to appear, December 1991.

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

[21] E. A. Riskin and R. M. Gray, "A Greedy Tree Growing Algorithm for the Design of Variable Rate Vector Quantizers ," *IEEE Trans. Signal Processing*, to appear Nov. 1991.

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

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

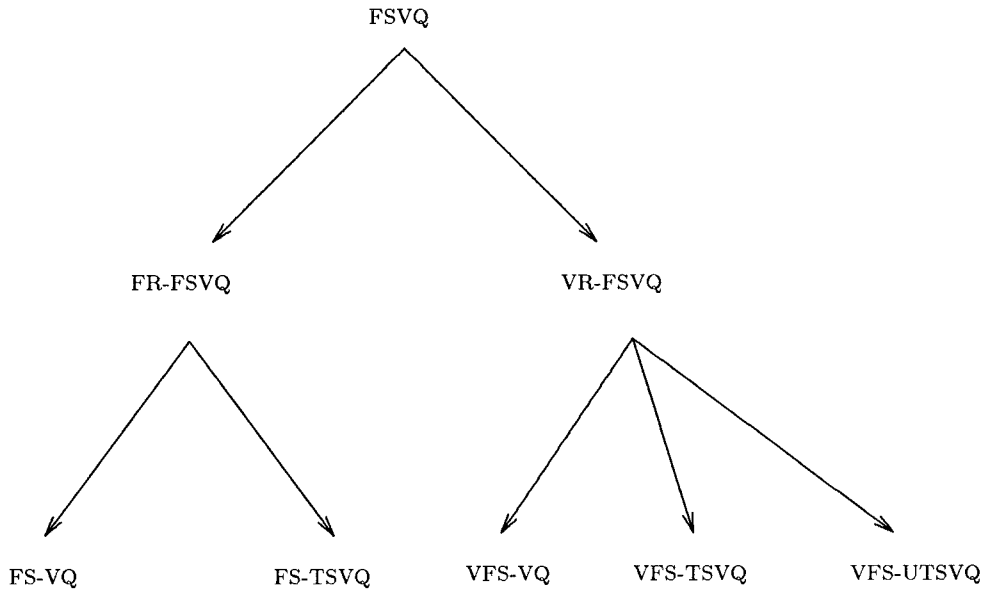


Figure 1: Notation tree.

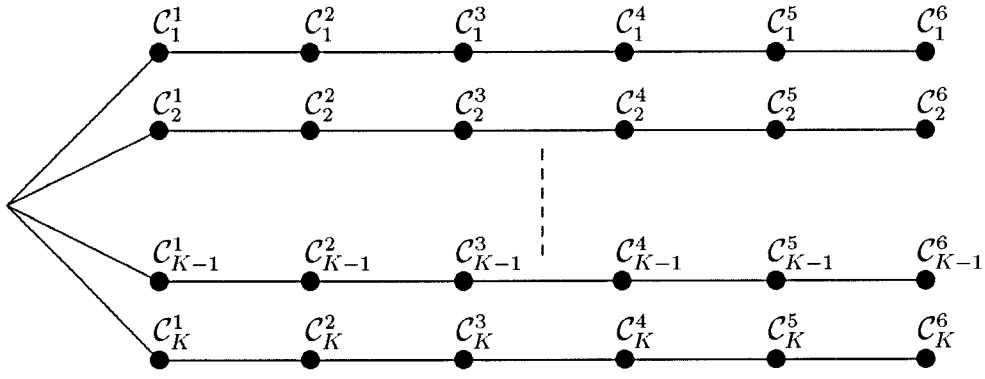


Figure 2: Tree used for bit assignment; $M_k = 6$ for all states.

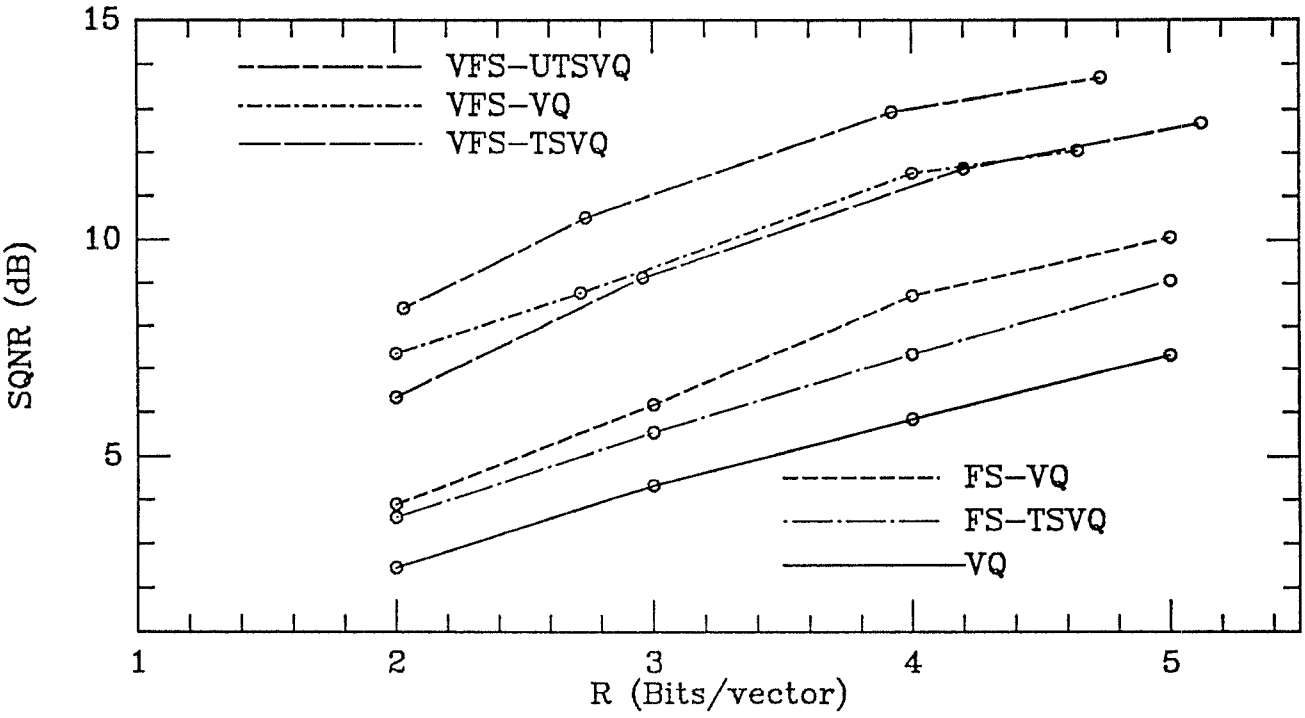


Figure 3: Performance of FR-FSVQs and VR-FSVQs on training sequence; $K = 32$.

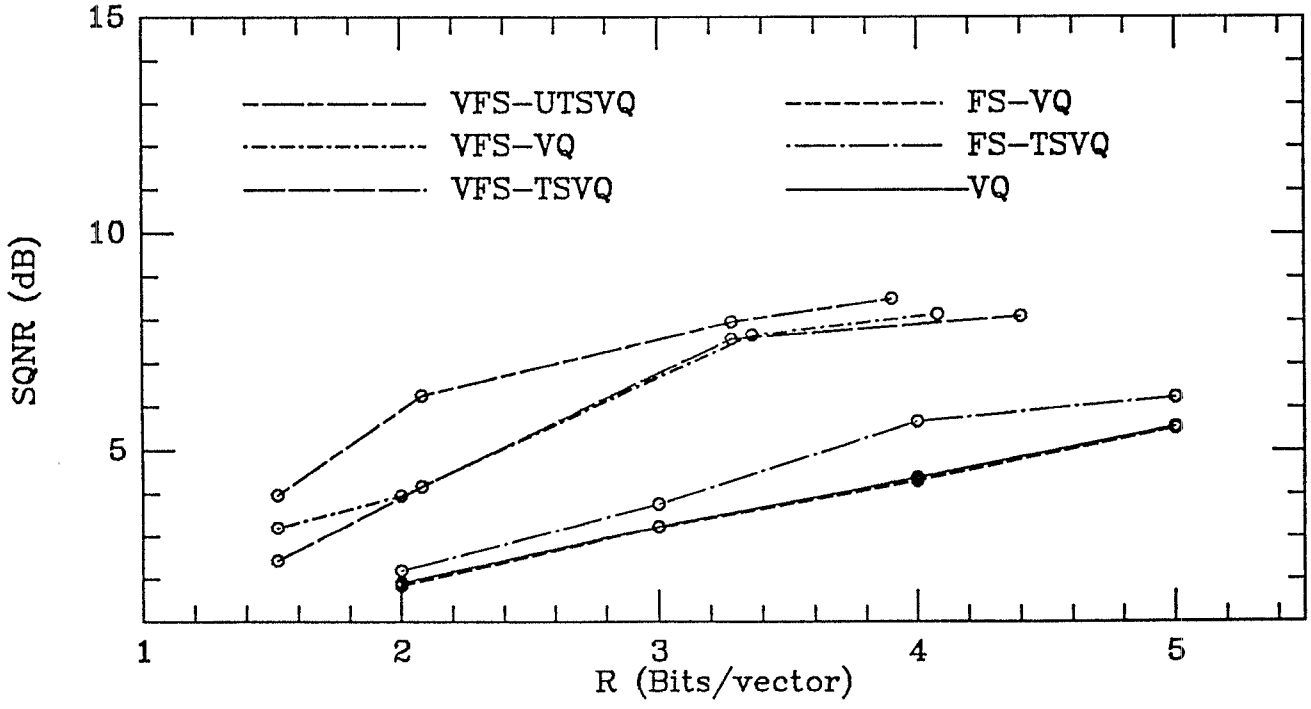


Figure 4: Performance of FR-FSVQs and VR-FSVQs on out-of-training sequence; $K = 32$.

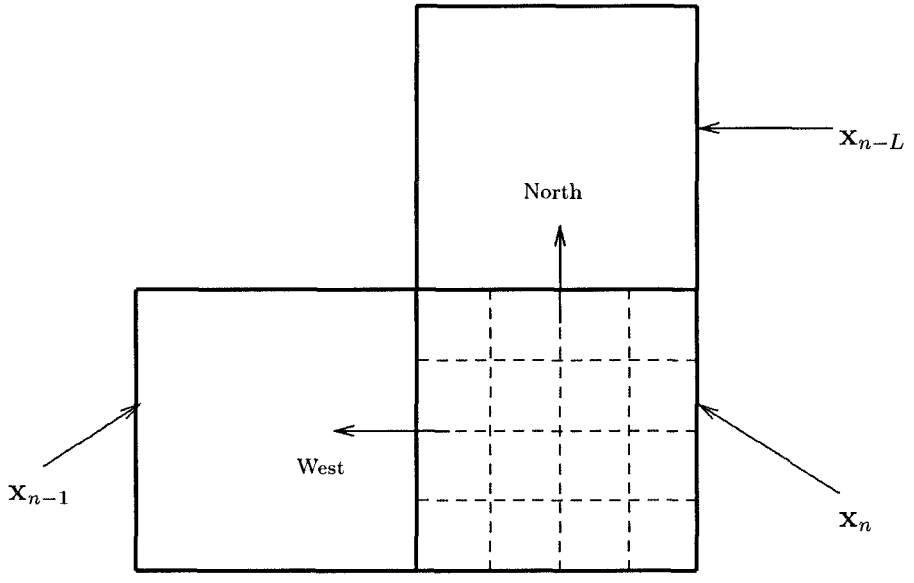


Figure 5: Input vector $\mathbf{x}_n$ and its north ($\mathbf{x}_{n-L}$) and west ($\mathbf{x}_{n-1}$) neighbors.

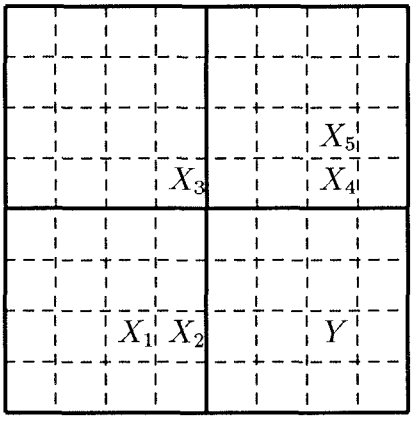


Figure 6: The predictor structure.

(a)

(b)

(c)

Figure 7: (a) Orignal, (b) ME-VFS-UTSVQ2 ($b = 0.31$), (c) PR-VFS-UTSVQ2 ($b = 0.32$).

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

Table 1: Performance of FS-VQ and VQ at $b = 0.375$, 0.5 and 0.625 bits/sample on the Synthetic Source.

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

Table 2: Performance of FS-TSVQ and TSVQ at $b = 0.375$, 0.5 and 0.625 bits/sample on the Synthetic Source.

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

Table 3: Performance of VFS-VQ at $b = 0.375$, 0.5 and 0.625 bits/sample on the Synthetic Source. Numbers in parentheses denote actual bit rate.

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

Table 4: Performance of VFS-TSVQ at $b = 0.375$, 0.5 and 0.625 bits/sample on the Synthetic Source. Numbers in parentheses denote actual bit rate.

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

Table 5: Performance of VFS-UTSVQ at $b = 0.375$, 0.5 and 0.625 bits/sample on the Synthetic Source. Numbers in parentheses denote actual bit rate.

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

Table 6: Performance of FS-VQ and VQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence.

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

Table 7: Performance of FS-TSVQ and TSVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence.

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

Table 8: Performance of VFS-VQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence. Numbers in parentheses denote actual bit rate.

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

Table 9: Performance of VFS-TSVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence. Numbers in parentheses denote actual bit rate.

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

Table 10: Performance of VFS-UTSVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence. Numbers in parentheses denote actual bit rate.

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

Table 11: Performance of FS-VQ and VQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence.

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

Table 12: Performance of FS-TSVQ and TSVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence.

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

Table 13: Performance of VFS-VQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence. Numbers in parentheses denote actual bit rate.

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

Table 14: Performance of VFS-TSVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence. Numbers in parentheses denote actual bit rate.

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

Table 15: Performance of VFS-UTSVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence. Numbers in parentheses denote actual bit rate.

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

Table 16: Performance of ME-VQ, ME-FS-VQ, ME-VFS-UTSVQ1 and ME-VFS-UTSVQ2 at $b = 0.19,\ 0.25,\ 0.31$ and 0.38 bits/pixel on the 512×512 version of Lena. Numbers in parentheses denote actual bit rate.

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

Table 17: Performance of PR-VQ, PR-FS-VQ, PR-VFS-UTSVQ1, PR-VFS-UTSVQ2 and RDG at $b = 0.20,\ 0.26$, and 0.32 bits/pixel on the 512×512 version of Lena. Numbers in parentheses denote actual bit rate.