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

Variable-Rate Finite-State Vector Quantization

by Y. Hussain and N. Farvardin

TR 90-22

Variable-Rate Finite-State Vector Quantization[†]

Y. Hussain and N. Farvardin
Electrical Engineering Department
Institute for Advanced Computer Studies
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 codeword selected determine the next encoder state [1]. In [1], the state codebooks are unstructured. In addition, it is assumed that all the state codebooks have the same cardinality leading to a fixed-rate system. In this paper, we present two *variable-rate* variations of the system in [1]. In the first system we let the state codebook sizes be different for different states. In the second system along with the flexibility of having different codebook sizes for different states, we use pruned tree-structured vector quantizers [2] as the state quantizers. For encoding sampled speech data, both of these schemes perform significantly better than the scheme in [1]. The second system gives the best performance of all. Performance improvements of up to 4.25 dB at the rate of 5/8 bits per sample are obtained.

[†]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.

1 Introduction

In the last decade an extensive amount of work has been done on vector quantization (VQ) as a means for data compression [3]-[6]. The main motivation behind the use of VQ was the result by Shannon 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 assumes no memory between successive samples. On the other hand, VQ assumes memorylessness between successive vectors. The performance of VQ can be further improved by either increasing the blocklength or exploiting the correlation between successive vectors. The option of increasing the blocklength is not very attractive due to the complexity issue, leaving us with the second option. In order to exploit the inter-vector correlation, several VQ systems with memory have been introduced in the literature [1], [3], [7], [8]. The so-called finite-state vector quantizer (FSVQ) [1], belongs to the class of VQs with memory. An FSVQ is a finite-state machine used for data compression. It can be viewed as a collection of memoryless VQs, where each input vector is encoded using a VQ associated with the current encoder state; the current state and codeword selected determine the next encoder state. FSVQ has been successfully applied to both speech and image coding and it is shown to perform substantially better than memoryless VQ [1],[8]-[11].

In the FSVQ systems described in [1], 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 much better. In one of the schemes described in this paper, we have relaxed this assumption as a result of which performance improvements (in some cases substantial) in terms of signal-to-quantization-noise ratio are obtained. We have also considered the possibility of using structured VQs such as tree-structured VQ (TSVQ) and pruned tree-structured VQ (PTSVQ) as the state quantizers. In fact, the scheme using PTSVQ as state VQs 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.

The rest of the paper is organized as follows. In Section 2, fixed-rate FSVQ systems as well as optimally pruned TSVQ systems are briefly discussed. Section 3 describes variable-rate FSVQ systems, while Section 4 provides the simulation results. Finally summary and conclusions are given in Section 5.

2 Preliminaries

In this section, we briefly provide the description and design algorithm of FSVQ with both structured and unstructured state codebooks. Also, we briefly describe the idea of optimally pruned TSVQ of [2] which will be the key to the systems to be described in the next section.

2.1 Finite-State Vector Quantizer and Finite-State Tree-Structured Vector Quantizer

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

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

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

(3) f: $\mathcal{N} \times \mathcal{S} \to \mathcal{S}$: 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:

$$\begin{aligned} u_n &= \alpha(\mathbf{x}_n, s_n), && (1) \\ \hat{\mathbf{x}}_n &= \beta(u_n, s_n), && (2) \\ s_{n+1} &= f(u_n, s_n), \quad n = 0, 1, \ldots. && (3) \end{aligned}$$

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 FSVQ. The rate of an FSVQ 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 FSVQ. The distortion measure $d : \mathbb{R}^L \times \hat{\mathcal{A}} \to [0,\infty)$ assigns a nonnegative 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

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

An FSVQ can be interpreted as a set of K full-searched VQs (one VQ associated with each state), each of codebook size N [3], [4]. The current input vector is vector-quantized using the VQ associated with the current state of the system; the current state and channel symbol determine the next state.

A finite-state tree-structured vector quantizer (FS-TSVQ) is specified in a manner very similar to FSVQ. Associated with each state, we now have a TSVQ rather than a full-searched VQ. The encoding is accordingly done in a tree-structured manner. In particular, the encoder mapping $\alpha : \mathbb{R}^L \times \mathcal{S} \to \mathcal{N}$ for FS-TSVQ differs from that of the FSVQ; 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. Next, we present the basic design algorithms of the FSVQ (FS-TSVQ) as given in [1].

2.2 FSVQ (FS-TSVQ) Design Algorithm

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

(a) the state codebooks $\mathcal{C}_k$ for FSVQ (FS-TSVQ), 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.

1. Using the LBG algorithm [4], design an ordinary memoryless VQ 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 FSVQ (FS-TSVQ), design an initial reproduction codebook $\mathcal{C}_k = \{\beta(u,k), u \in \mathcal{N}\}$ using the LBG algorithm [4] on the subtraining sequence composed of all successors to vectors for which the state-label VQ chooses k, 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 FSVQ (FS-TSVQ) 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, 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

$$f(u,k) = \arg\min_{s\in\mathcal{S}} d(\beta(u,k), \mathbf{c}(s)),\ k \in \mathcal{S}, u \in \mathcal{N}. \qquad (4)$$

4. Attempt to improve the state codebooks $\{\mathcal{C}_k, k \in \mathcal{S}\}$ of the FSVQ (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 state-label VQ $\mathcal{C}$ similarly.

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 FSVQs designed with this algorithm exhibit substantial gain over ordinary memoryless VQs in both waveform coding and vocoding applications [1], [8].

In the FSVQ 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 FSVQ, the cardinality of each state codebook is assumed to be 2^R. In the next section, we drop this assumption and present a modified

FSVQ system (and also a modified FS-TSVQ system), in which the state codebook sizes are optimally chosen; this leads to performance gains (in some cases substantial). We also consider a second variation of the FSVQ system in which optimally pruned TSVQs [2] are used as state VQs and state codebook sizes are chosen optimally.

In the following, we provide a brief description of the optimally pruned tree-structured vector quantizers described in [2] which is the key to the systems to be discussed.

2.3 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 a pruned TSVQ (PTSVQ) is obtained. The codebook of the PTSVQ 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 PTSVQ with average rate $\ell(\mathcal{P})$ and average distortion $\delta(\mathcal{P})$. The operational distortion-rate function defined by

$$\hat{D}(R) = \min_{\mathcal{P} \preceq \mathcal{T}} \{\delta(\mathcal{P}) / \ell(\mathcal{P}) \leq R\} \tag{5}$$

specifies the optimal trade-off between rate and distortion over all pruned subtrees of $\mathcal{T}$. An algorithm is presented in [2] which traces out the convex-hull of the operational distortion-rate function. The algorithm given in [2] 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 if 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 Variable-Rate FSVQ

In this section, we present the modified FSVQ systems. We begin with the description of the variable-rate FSVQ systems.

3.1 Description of Variable-Rate FSVQ and FS-TSVQ

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. The modified versions of FSVQ and FS-TSVQ will be subsequently referred to as V-FSVQ and V-FS-TSVQ, respectively. Both V-FSVQ and V-FS-TSVQ can again be 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) β: $\mathcal{N}(\mathcal{S}) \times \mathcal{S} \rightarrow \hat{\mathcal{A}}$: finite-state decoder,
(3) f: $\mathcal{N}(\mathcal{S}) \times \mathcal{S} \rightarrow \mathcal{S}$: next state function.

Here the channel alphabet depends of 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 $R = \sum_{k=1}^{K} P_k \log_2 N_k$, bits/vector, where N_k is the cardinality of $\mathcal{N}_k$ and P_k is probability of occurrence of state k. Now since the bit rates associated with different state codebooks are not constrained to be the same, state codebook sizes become an additional set of variables in the design stage. In the next subsection, we provide a description of the method used to determine the state codebook sizes.

3.2 Bit 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, $D_{k,R_{k,1}} \geq D_{k,R_{k,2}} \geq \cdots \geq D_{k,R_{k,M_k}}$, $\forall k \in \mathcal{S}$. 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 FSVQ, 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

$$D = \sum_{k=1}^{K} P_k D_{k,b_k}, \tag{6.a}$$

subject to

$$\sum_{k=1}^{K} P_k b_k \leq b_{avg}, \tag{6.b}$$

and

$$b_k \in \{R_{k,1}, R_{k,2}, \ldots, R_{k,M_k}\}. \tag{6.c}$$

The above bit assignment problem can be solved using the idea of optimal pruning of a TSVQ. In that, we first construct the following tree $\mathcal{T}$: The root node of $\mathcal{T}$ 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 variable-rate FSVQ 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 variable-rate FSVQ associated with $\mathcal{P}$ is given by

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

and the average distortion is given by

$$\delta(\mathcal{P}) = \sum_{k=1}^{K} P_k D_{k,R_{k,l_k}}. \tag{8}$$

The optimal pruning algorithm of [2], 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 $\mathcal{P} \preceq \mathcal{T}$.

3.3 V-FSVQ (V-FS-TSVQ) Design Algorithm

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

$$D = \sum_{k=1}^{K} P_k D_{k,b_k}, \tag{9}$$

subject to a constraint on the average rate described by

$$\sum_{k=1}^{K} P_k b_k \leq b_{avg}, \tag{10}$$

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 V-FSVQ (V-FS-TSVQ), 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 [4].

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 ordinary VQs (TSVQs) of rates $1, 2, \ldots, b_{max,k}$[1] bits/vector.

[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.

We denote the set of VQs and TSVQs 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. 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 V-FSVQ (V-FS-TSVQ).

4. As in the case of FSVQ (FS-TSVQ), the next-state function f is defined as

$$\begin{aligned} f(u,k) = \ & \arg\min_{s \in \mathcal{S}} d(\beta^{b_k^*}(u,k), \mathbf{c}(s)), \\ & k \in \mathcal{S}, u \in \{1, 2, \ldots, 2^{b_k^*}\}. \end{aligned} \tag{11}$$

5. Encode the whole training sequence using the next-state function f and the state codebooks $\{\mathcal{C}_k^{b_k^*}, k \in \mathcal{S}\}$ for V-FSVQ (V-FS-TSVQ). 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.a (for V-FSVQ) $\forall k \in \mathcal{S}$, 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.a) for some predetermined number of iterations.

6.b (for V-FS-TSVQ) Using the training sequences obtained in step (5), repeat steps (2), (3), (4) and (5) for some predetermined number of iterations.

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

3.4 Description and Design of Variable-Rate Finite-State Pruned Tree-Structured Vector Quantizer

In another variation of the FSVQ scheme, we consider a system in which the state quantizers are optimally pruned TSVQs obtained using the algorithm in [2]. 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 TSVQ over full-search VQ along with the additional advantage of fast encoding due to the tree-searched method. As a result of using PTSVQs as state VQs, even for a given state the encoding rate is variable with time now, while for V-FSVQ (V-FS-TSVQ) the rate varies between states but is fixed within each state. We refer to the new scheme as V-FS-PTSVQ. The system is formally described in the same way as V-FS-TSVQ. The design algorithm is also similar to that of V-FS-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 V-FSVQ (V-FS-TSVQ). Then, using the optimal pruning algorithm of [2] on each of the complete state TSVQs, we obtain K setsof collection of optimally pruned TSVQs. The rate of the optimally pruned TSVQs associated with state k varies from 1 bit/vector to $\hat{b}_{max,k}$ bits/vector and takes finitely many discrete values which are not necessarily integers; the fact that rates are not constrained to be integers as in V-FSVQ (V-FS-TSVQ) leads to an additional improvement factor. Then we apply the bit assignment algorithm on the collection of K sets of optimally pruned TSVQs 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 V-FS-TSVQ. Again, the algorithm is suboptimal but it gives substantial gain over all other schemes considered in this paper.

4 Simulation Results

We performed extensive simulations to compare the variable-rate FSVQ systems described in this paper with the scheme described in [1]. The performance measure used is the signal-to-quantization-noise ratio (SQNR). We denote by b the average bit rate per sample. FSVQs were designed for speech waveform with vector dimension $L = 8$ and $K = 8$, 16 and 32. The database used for these designs was a sequence of speech samples consisting of five minutes of speech sampled at 8 KHz and uttered by five male and three female speakers.

The performance of FSVQ [1] for $b = 0.25, 0.375, 0.5$ and 0.625 bits/sample is illustrated in Table 1. Also for the sake of comparison, we include the performance results of ordinary VQ in terms of SQNR. As claimed in [1], the FSVQ system outperforms the VQ system by over 2 dB and the gain increases with the number of states.

Table 2 illustrates the performance of FS-TSVQ system in comparison with the VQ system. The trend is similar to that of FSVQ. Comparison of Tables 1 and 2 shows that under similar conditions, FS-TSVQ performs slightly worse than the FSVQ system since there is a degradation in the performance when we replace a full-searched VQ by a complete TSVQ of the same rate. In most cases, FS-TSVQ system yields a SQNR within 1.0 dB of the FSVQ.

Table 3 summarizes the performance of V-FSVQ for different values of b and K. The term in the braces is the value of the bit rate per sample actually achieved by the system, while the desired value is given by b. Comparison of Table 1 and 3 indicates that at the same bit rate, V-FSVQ outperforms FSVQ system, 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 algorithm in [2] is small.

The performance of V-FS-TSVQ is illustrated in Table 4. It should be noted that the difference between the performance of V-FS-TSVQ and V-FSVQ is smaller than the difference between FS-TSVQ and FSVQ. The reason resides in the limitation in the size of the largest full-searched VQ (2048 codevectors in codebook) needed for V-FSVQs; this limitation is less severe for TSVQs.

Finally, we include the performance results of V-FS-PTSVQ in Table 5. This system gives the best performance results among all the schemes considered in this paper. Again, the term in the braces represents the actual achieved rate. For all values of K considered here, V-FS-PTSVQ performs better than the FSVQ by at least 3 dB and by as much as 4.25 dB for $b = 0.5$ bits/sample and higher.

The performance of V-FSVQ and V-FS-PTSVQ schemes on an out-of-training test sequence is summarized in Tables 7 and 8. The test sequence was 67 seconds of speech sampled at 8 KHz. It consisted of several sentences spoken by a male speaker. The performance results of VQ and FSVQ schemes are also included for comparison. Study of these tables shows that variable-rate FSVQs outperform fixed-rate FSVQ and VQ. The SQNR gains for V-FS-PTSVQ, when b is 0.375 bits/sample and higher is close to 2 dB while the actual rate of the variable-rate FSVQ systems is much lower than b. Comparison at the same bit rate shows a gain of as much as 3 dB for V-FS-PTSVQ and 2.4 dB for V-FSVQ over the fixed-rate FSVQ.

5 Summary and Conclusion

We have considered two *variable-rate* variations of the FSVQ system described in [1]. The design algorithms for the variable-rate FSVQs are obtained by simple modification of the FSVQ design algorithm. None of these algorithms (including the one described in [1]) converge and there is no guaranteed improvement at each step of iteration.

The variable-rate FSVQ systems considered in this paper perform substantially better than the system in [1] and SQNR gains of up to 4.25 dB on in-training sequence and 3 dB on out-of-training test sequence are obtained. We believe that higher gains can be achieved even on the test sequence outside the training sequence if sufficiently long sequence is used to train the variable-rate FSVQ systems.

One shortcoming of the variable-rate FSVQ systems is that if it is designed to achieve a certain encoding rate for a training sequence, it may give a different rate if used to encode another sequence. To circumvent this difficulty, some type of algorithm which adaptively modifies the encoding system based on some local (instantaneous) measurement of the encoding rate is needed.

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] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] L.C. Stewart, R.M. Gray and Y. Linde, "The Design of Trellis Waveform Coders," *IEEE Trans. Commun.*, vol. COM-30, pp. 702-710, April 1982.

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

[10] C.S. Kim, J. Bruder, M.J.T. Smith and R.M. Mersereau, "Subband Coding of Color Images Using Finite State Vector Quantization," *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, pp. 753-756, April 1988.

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

	FSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
M	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 1: Performance of FSVQ and VQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence.

	FS-TSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
M	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

Table 2: Performance of FS-TSVQ at $b = 0.25$, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence.

	V-FSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
M	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 (0.385)	10.85 (0.52)	12.02 (0.64)
32	7.34 (0.25)	8.76 (0.34)	11.52 (0.50)	12.04 (.58)

Table 3: Performance of V-FSVQ at $b =$ 0.25, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence.

	V-FS-TSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
M	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 4: Performance of V-FS-TSVQ at $b =$ 0.25, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence.

	V-FS-PTSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
M	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)	9.68 (0.375)	12.61 (0.49)	13.59 (0.625)
32	8.31 (0.25)	9.93 (0.375)	12.92 (0.49)	13.58 (0.59)

Table 5: Performance of V-FS-PTSVQ at $b =$ 0.25, 0.375, 0.5 and 0.625 bits/sample on the Training Sequence.

	FSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
M	SNR	SNR	SNR	SNR
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 6: Performance of FSVQ and VQ at b = 0.25, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence.

	V-FSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
M	SNR	SNR	SNR	SNR
8	2.08 (0.17)	2.50 (0.21)	7.27 (0.39)	7.88 (.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 (.51)

Table 7: Performance of V-FSVQ at b = 0.25, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence.

	V-FS-PTSVQ			
	$b = .25$	$b = .375$	$b = .5$	$b = .625$
M	SNR	SNR	SNR	SNR
8	2.79 (0.18)	5.43 (0.32)	7.44 (0.40)	7.95 (0.525
16	3.02 (0.18)	5.24 (0.32)	7.89 (0.41)	8.44 (0.54)
32	3.31 (0.18)	4.58 (0.25)	7.96 (0.41)	8.43 (0.49)

Table 8: Performance of V-FS-PTSVQ at b = 0.25, 0.375, 0.5 and 0.625 bits/sample on Out-of-Training Test Sequence.