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 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

A Structured Fixed-Rate Vector Quantizer Derived from a Variable-Length Scalar Quantizer

by R. Laroia and N. Farvardin

TR 91-76

A Structured Fixed-Rate Vector Quantizer Derived from a Variable-Length Scalar Quantizer †

by

Rajiv Laroia and Nariman Farvardin
Electrical Engineering Department and
Systems Research Center
University of Maryland
College Park, Maryland 20742

Abstract

The well-known error propagation problem inherent in any variable-length coding operation limits the usefulness of variable-length encoded scalar quantizers for transmission over noisy channels. In the absence of channel noise however, these quantizers are known to perform better than error-minimizing fixed-rate Lloyd-Max quantizers for a wide class of memoryless sources. Motivated by this observation, in this paper we develop a low complexity fixed-rate structured vector quantizer in which the structure of the codebook is derived from a variable-length scalar quantizer. Consider an n-level variable-length (e.g., Huffman coded) scalar quantizer $\mathcal{S}$ quantizing samples from a memoryless stationary source. Out of all possible n^m m-vectors (m consecutive samples) at the output of $\mathcal{S}$, the 2^{mr} vectors with the smallest total (encoded) length are chosen as the codebook of an m-dimensional rate r bits/sample structured vector quantizer derived from $\mathcal{S}$. A procedure for the design of this quantizer along with fast algorithms for implementation are developed and bounds on the performance of the scheme are developed. The structured vector quantizer can be designed and implemented even for fine (high rate) quantization at relatively large block-lengths and can achieve a rate-distortion performance superior to that of implementable LBG vector quantizers. Numerical results demonstrating the efficacy of this scheme along with comparisons against Lloyd-Max quantizers and optimal entropy-constrained quantizers, both in the absence and presence of channel noise are rendered. These results show that performance close to that of optimal entropy-constrained scalar quantizers is possible with this fixed-rate quantizer. The structured vector quantizer is also robust against channel errors and outperforms both Lloyd-Max and entropy-constrained scalar quantizers for a wide range of channel error probabilities. Extension of this idea to other types of sources proves very useful and will be described in a sequel paper.

Index Terms: Fixed-rate coding; structured vector quantization; variable-length scalar quantization; memoryless sources.

---

† 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.

I. Introduction

Optimal entropy-constrained scalar quantizers (ECSQs) are known to perform close to the rate-distortion bound (within 0.25 bits/sample) for a large class of memoryless sources [1], [2]. To implement such a quantizer, however, some type of *variable-length* noiseless coding is needed. Due to the sequential nature of decoding of such codes, channel errors could lead to a loss of synchronization resulting in propagation of error and hence large cumulative reconstruction errors. Another problem with the ECSQ is that it requires an output buffer to convert its variable-rate output to a fixed-rate bit stream for transmission over a channel. In any practical situation the buffer is of finite size and hence undesirable buffer overflow/underflows can occur. The ECSQ is therefore of a limited usefulness for most practical transmission applications especially over noisy channels; usually some type of packetization to limit the error propagation to within a packet is used.

Optimal distortion minimizing Lloyd-Max quantizers (LMQs) utilizing *fixed-length* codewords, on the other hand, are immune to error propagation and related problems but in the absence of channel noise perform worse than the ECSQ [3],[4]. The gap between the performance of the optimal ECSQ and LMQ can be large for certain sources [2]. To bridge this gap while maintaining fixed-length codewords one could use a *fixed-rate* vector quantizer (VQ) [5], [6]. An optimal fixed-rate VQ of a sufficiently large dimension can perform arbitrarily close to the rate-distortion bound [7]. Such a quantizer however has three drawbacks: (i) the encoding complexity (which grows exponentially with the rate and block-length), (ii) the large amount of memory required to store the codebook and (iii) the amount of data required to train the VQ (which grows with the codebook size) limiting the size of the codebook that can be designed [5]. Various vector quantization schemes for reduced encoding complexity have been developed. Among others, the tree-searched VQ [8], the multi-stage VQ [9], the product VQ [10], the fine-coarse VQ [11] and the lattice VQ [12] can be mentioned. An expository treatment can be found in [13]. In general, these VQs reduce the encoding complexity at the cost of larger memory requirement and/or some performance degradation.

The permutation coder (PC) is a *fixed-rate* encoder which can be thought of as a VQ with a special structure (the codebook consists of codevectors that are permutations of one another) [14]-[16]. The structure of this codebook obviates the need for storing the codevectors and also results in a simple quantization algorithm. It is proved in [15] that the

average distortion of the PC is bounded below by that of the optimal ECSQ at the same rate and in the limit of infinite block-length, the optimal PC becomes equivalent to the optimal ECSQ. The block-length of the PC required to achieve a performance better than LMQ is very large resulting in large encoding delays. In addition, while the channel errors do not propagate beyond block boundaries, they could affect all samples within a block; therefore very large block-lengths offset some of the advantages of fixed-length coding over noisy channels.

The pyramid vector quantizer (PVQ) introduced by Fischer for Laplacian sources [17] is also a *fixed-rate* VQ in which the codevectors are placed on the intersection of a cubic lattice and a pyramid in m-dimensional space where m is the block-length of the PVQ. While for Laplacian sources this quantizer is asymptotically optimal (in block-length m) and achieves the performance of the ECSQ, for other sources it does not approach the ECSQ performance even for large m.

The *fixed-rate* spherical coordinate quantizer [18], [19] can efficiently quantize spherically symmetric sources by representing source vectors in spherical coordinates and then using scalar quantizers to quantize the coordinate values. Most memoryless symmetric sources however (with the exception of Gaussian sources) do not exhibit spherical symmetry. These quantizers will therefore not quantize such sources very efficiently.

This paper describes a fixed-rate *structured vector quantizer* (SVQ) for a memoryless stationary source that bridges the gap between optimal ECSQ and LMQ but is significantly less complex than a VQ. It is shown in [20] that the SVQ can be extended to other types of sources also. The basic idea can be described as follows. Consider an ECSQ the outputs of which are encoded by variable-length codewords of length given by the negative logarithm of the probability[1] of the corresponding quantization level. If such a variable-length encoded ECSQ is used to quantize a block of samples from a memoryless source, the total length of the output reflects its probability (smaller length means higher probability). When the block-length is large, with a high probability, the length of the output is close to the typical length $\equiv$(*block-length* $\times$ *entropy*). This suggests that if (for large block-lengths) the output length of a variable-length encoded ECSQ is constrained to lie in a small neighborhood of the typical length, then with high probability the quantizer outputs can be encoded without any distortion. If all the possible ECSQ outputs that satisfy the

[1] For simplicity assume that this leads to integer lengths.

length constraint are counted and encoded with fixed-length codewords, the system will perform close to the ECSQ. Although the performance of this system approaches that of the ECSQ as the block-length goes to infinity, for any fixed block-length it is preferable to encode all the ECSQ outputs of length equal to or less than the typical length. This is because the outputs with smaller length are more probable. This idea provides the motivation for the present work and leads to the definition of the SVQ.

It is shown that the SVQ is implementable even for very large codebooks and successfully bridges the performance gap between the LMQ and the optimal ECSQ while maintaining fixed-length outputs. Furthermore, it is demonstrated that, in certain cases, the SVQ outperforms the best implementable LBG VQ [5]. For transmission over noisy channels, the SVQ generally performs better than both ECSQ and LMQ.

The rest of this paper is organized as follows. A formal definition of the SVQ is provided in Section II followed by SVQ design algorithms in Section III. Section IV addresses the implementation issues which include codebook search and codevector encoding. Bounds on the performance of the SVQ and comparisons with other schemes are made in Section V while the storage and the computational requirements of the SVQ are analyzed in Section VI. Specific numerical results demonstrating the efficacy of this scheme for a variety of sources, encoding rates and block-lengths are presented in Section VII including comparisons with the LMQ, ECSQ, PC and VQ. Numerical results in the presence of channel noise are given in Section VIII followed by a summary and conclusions in Section IX.

II. Structured Vector Quantizer - Preliminaries and Definition

In the sequel we will assume that the source to be encoded is a discrete-time, stationary *memoryless* source, $\{X_n\}$, with a probability density function (pdf) $p(x)$. We use $d(x,y)$ to denote a non-negative, single-letter, distortion measure between the source output x and its reproduction y.

An N-level, m-dimensional VQ is described in terms of a codebook $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N\}$ where $\mathbf{c}_i \in \mathbb{R}^m, \forall i \in J_N \equiv \{1, 2, \ldots, N\}$, in such a way that an input vector $\mathbf{x} \in \mathbb{R}^m$ is quantized to $\mathbf{c}_i$ if $d(\mathbf{x}, \mathbf{c}_i) \leq d(\mathbf{x}, \mathbf{c}_j), \forall j \in J_N$, [5]. A fixed-rate implementation of this VQ corresponds to an encoding rate of $r = (\log_2 N)/m$, bits/sample.

The *structured vector quantizer* (SVQ) is a specific kind of VQ in which the codebook structure is derived from a variable-length scalar quantizer (SQ). The objective is

to construct a *fixed-rate* VQ whose rate-distortion performance for a memoryless source is close to that of the optimal (*variable-length* encoded) ECSQ. To describe the dependence of the SVQ on the variable-length SQ, consider an n-level SQ $\mathcal{S}$ described in terms of the set of quantization levels[2] $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$ and the corresponding set of *lengths* $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_n\}$, where the ℓ_i's are integers. For the present discussion, the length ℓ_i, $i \in J_n$, is assumed to be the length of the binary codeword corresponding to q_i if a variable-length code (e.g., Huffman code) is used at the output of $\mathcal{S}$. We will soon use the ℓ_i's in a more general sense for constructing the SVQ. If each component of an input m-vector (block of m input samples) is separately quantized using $\mathcal{S}$, the quantized vector is in $\mathcal{Q}^m$ which is an m-dimensional grid of n^m points. The codebook of the SVQ $\mathcal{V}$ derived from $\mathcal{S} \equiv (\mathcal{Q}, \mathcal{L})$ is a subset of $\mathcal{Q}^m$. Specifically, if $\mathcal{V}$ is a rate r bits/sample SVQ, a vector in $\mathcal{Q}^m$ is a codevector of $\mathcal{V}$ only if its total length (defined as the sum of the lengths of its components) is no greater than a threshold L, where L is chosen such that the codebook contains no more than 2^{mr} codevectors. A formal definition of the SVQ now follows.

Definition 1: An m-dimensional SVQ $\mathcal{V}$ derived from the variable-length encoded SQ $\mathcal{S} = (\mathcal{Q}, \mathcal{L})$ is a VQ with a codebook $\mathcal{Z}$ given as,

$$\mathcal{Z} = \{\mathbf{z} \equiv (z_1, z_2, \ldots, z_m) \in \mathcal{Q}^m \ : \ \ell_{f(z_1)} + \ell_{f(z_2)} + \cdots + \ell_{f(z_m)} \leq L\}, \tag{1}$$

where the index function $f : \mathcal{Q} \rightarrow J_n$ is defined as,

$$f(q_i) = i, \ i \in J_n \ . \tag{2}$$

For a rate r bits/sample SVQ, the threshold L is selected as the largest integer such that $\text{card}(\mathcal{Z}) \leq 2^{mr}$.

Note that an m-dimensional SVQ $\mathcal{V}$ is completely defined in terms of the triple $(\mathcal{Q}, \mathcal{L}, L)$, i.e., the quantization levels of the SQ, the set of lengths associated with these levels and a threshold. We will show in Section IV that this simple structure of the SVQ codebook results in fast and efficient algorithms for codebook search and encoding.

[2] To completely describe the quantizer $\mathcal{S}$, its quantization thresholds also need to be specified. This is omitted here because the thresholds are not used in defining the SVQ.

Discussion:

The following argument is provided to describe qualitatively the motivation behind the SVQ as well as its relationship with the variable-length encoded SQ, the permutation coder and the pyramid vector quantizer.

Let $\mathcal{S} = (\mathcal{Q}, \mathcal{L})$ together with the quantization thresholds denote a variable-length encoded ECSQ for which the average distortion is minimized while the output entropy is equal to a prescribed value, say H [2], [15]. The length ℓ_i of the quantization level q_i is assigned according to $\ell_i = \log(1/p_i)$, where p_i is the probability of quantizing the input sample to q_i. For simplicity assume that the above length assignment results in integer lengths[3]. The average length of the encoded quantizer output now equals the entropy H.

If a source m-vector (large m) is quantized using $\mathcal{S}$, the weak law of large numbers implies that the total length of the encoded output vector will be close to mH with a very high probability. Also, we know that the probability of an encoded sequence of length mH at the output of $\mathcal{S}$ equals 2^{-mH}. This suggests that choosing output m-vectors that are encoded to a length in a small neighborhood of mH identifies a set of (almost) equi-probable vectors and the total probability of this set is close to one. Hence there are approximately 2^{mH} of such vectors and these can be encoded by a fixed-length code of rate H bits/sample. It is not difficult to show that for a large m there are relatively few output vectors that are encoded to lengths smaller than mH and hence including these in the above set does not significantly alter the rate of the fixed-length code. Hence, an SVQ derived from $\mathcal{S}$ with the threshold $L = mH$ has a rate-distortion performance (for large m) close to that of the ECSQ.

The permutation coder (PC) is another fixed-rate vector quantizer based on similar ideas. All codevectors of the PC are permutations of a given vector and have the same *type*[4]. This corresponds to a subset of m-vectors at the output of $\mathcal{S}$ that are encoded to the same length. It is shown in [15] that the performance of the PC, for large m, approaches that of the scalar quantizer $\mathcal{S}$ from which it is derived. Hence the performance of the PC derived from the optimal ECSQ approaches the performance of the optimal ECSQ. In practice, however, large values of m are needed to get satisfactory performance from PCs

[3] This is not really a constraint because later we will show that the SVQ can be described in terms of non-integer lengths.

[4] The type of a vector is the observed (probability) distribution of the elements of the source alphabet in the vector.

[16]. The SVQ can be thought of as a modification of the PC where vectors with types that are more probable (smaller total length) than the *typical* vectors are also included in the codebook. We will demonstrate later that for relatively small values of m this modification results in performance superior to that of the PC. This is because for small m there is no single type for which the set of all vectors of that type has an appreciable probability.

So far the set $\mathcal{L}$ was assumed to be the set of codeword lengths of a variable-length code. However, even if those lengths were arbitrary, Definition 1 can still be used to define an SVQ. Hence from now on we assume that the set $\mathcal{L}$ in Definition 1 can consist of arbitrary integer lengths. In fact, we do not even require these to obey Kraft's inequality. The requirement that they be integers is only for implementation reasons. While we have argued that the performance of an SVQ derived from an ECSQ approaches the performance of the ECSQ as block-length increases, at this point there is no obvious reason to believe that we cannot do better than the ECSQ if the SVQ is optimized over all $\mathcal{Q}$ and $\mathcal{L}$. However, we prove in Section V that the performance of the SVQ is indeed upper bounded by the performance of the optimal ECSQ.

The pyramid vector quantizer (PVQ) for Laplacian sources is also based on the idea of identifying a (close to) unit probability set of (almost) equi-probable m-vectors. However, in this case the set is identified at the source output (before quantization) and is a subset of $\mathbb{R}^m$. Vector quantization is performed assuming that the source output is in this set. The codebook here consists of the intersection of a cubic lattice with this unit probability set. For a stationary memoryless Laplacian source this set is a pyramid in $\mathbb{R}^m$ and hence the name pyramid vector quantizer. Fischer has shown that for Laplacian sources the performance of the optimal PVQ approaches that of the optimal ECSQ as the block-length increases. The codebook $\mathcal{Z}$ of the PVQ is of the form

$$\mathcal{Z} = \{\alpha \mathbf{i} \equiv (\alpha i_1, \alpha i_2, \ldots, \alpha i_m) : \mathbf{i} \in J^m \text{ and } |i_1| + |i_2| + \cdots + |i_m| = I\}, \qquad (3)$$

where J is the set of integers, α is a real constant and I is a positive integer. The PVQ can be thought of as an SVQ where the quantization levels are uniformly spaced, i.e., $\mathcal{Q} = \{0, \pm\alpha, \pm 2\alpha, \ldots, \pm n\alpha\}$, the lengths are given by $\ell_i = |q_i|/\alpha$ and the threshold $L = I$. The only difference is that for the PVQ only lattice points on the pyramid are chosen as codevectors while for the corresponding SVQ all points inside the pyramid are also included in the codebook. As mentioned before, including the points inside the pyramid does not affect the rate when m is large and hence the two quantizers are asymptotically

equivalent. Fischer's spherical vector quantizer [21] for Gaussian sources is defined along the same lines. While Fischer's approach of identifying unit probability sets works for Laplacian and Gaussian sources because it leads to simple geometrical shapes, it is in general not easy to do this for sources with arbitrary distribution. The PVQ is however used to quantize other sources; the asymptotic performance in these cases is below that of optimal ECSQ [17]. As we shall show in Section V, the optimal SVQ asymptotically achieves the optimal ECSQ performance for all sources.

The question that naturally arises now is how to pick $\mathcal{Q}$, $\mathcal{L}$ and L so as to minimize the average distortion of the SVQ at a given rate. This issue is addressed in the next section.

III. Design of the SVQ

An m-dimensional SVQ is completely described by $\mathcal{Q}$, $\mathcal{L}$ and L. Designing the SVQ hence corresponds to the determination of these quantities. A reasonable way to design the SVQ is to derive it from an ECSQ, i.e., choose $\mathcal{Q}$ and $\mathcal{L}$ of the SVQ as the quantization levels and the codeword lengths, respectively, of an optimally designed n-level ECSQ of rate (entropy) r and choose the threshold L such that the SVQ codebook has 2^{mr} vectors. Following the discussion in the previous section, one expects the performance of this SVQ to approach that of the corresponding ECSQ as the block-length m becomes large. Also, as shown in Section V, for large m the performance of the SVQ is upper bounded by the performance of the optimal ECSQ. Hence, the above procedure should lead to asymptotically (in block-length) optimal designs. While this procedure is expected to do well for very large values of m, as we will demonstrate in Section VII, superior performance for smaller (practical) values of m can be obtained. In what follows a two-step iterative procedure (similar to the Lloyd algorithm [3] for scalar quantizer design) for SVQ design is described and will be demonstrated to perform better.

The first step (Step A) of the two step procedure determines the quantization levels $\mathcal{Q}$ given $\mathcal{L}$ and L. The second step (Step B) determines the lengths $\mathcal{L}$ and threshold L given $\mathcal{Q}$. If both of these steps are performed optimally (with respect to some distortion measure), then each iteration reduces the average distortion and since the average distortion is lower bounded by zero, convergence to a locally optimal solution is guaranteed. While we have determined the optimal solution for Step A, unfortunately, the optimal solution for Step B (i.e., determining $\mathcal{L}$ and L given $\mathcal{Q}$) is not known; the unknowns in Step B are hence

determined suboptimally resulting in a suboptimal design of the SVQ. The algorithms used in Step A and Step B, as in conventional VQ design [5], use a long training sequence of N source vectors to characterize the source *pdf*. We denote by $\mathcal{X} = \{\mathbf{x}_j;\ j \in J_N\}$ the training set of source m-vectors $\mathbf{x}_j = (x_{1j}, x_{2j}, \ldots, x_{mj})$.

Step A and Step B of the design procedure are described next. For Step B two different suboptimal approaches are presented each giving rise to a different design algorithm.

A. Step A: Determination of $\mathcal{Q}$ given $\mathcal{L}$ and L

The problem of determining $\mathcal{Q}$ optimally given $\mathcal{L}$, L and a source *pdf* is itself solved using an iterative algorithm similar to that used for optimal design of conventional VQs [5]. The algorithm is described below.

Algorithm

0. For given $\mathcal{L}$, L, the training sequence $\mathcal{X}$ and stopping threshold $\epsilon > 0$, select the initial set of quantization levels $\mathcal{Q}_0 \equiv \{q_1^0, q_2^0, ..., q_n^0\}$ as the optimal solution for $\mathcal{Q}$ in the previous visit[5] to Step A, and set the iteration index $k = 0$.
1. Quantize each vector in $\mathcal{X}$ using the SVQ defined by $(\mathcal{Q}_k, \mathcal{L}, L)$. Let $\mathcal{Y}_k = \{\mathbf{y}_j^k;\ j \in J_N\}$ represent the resulting set of quantized vectors where $\mathbf{y}_j^k \equiv (y_{1j}^k, y_{2j}^k, \ldots, y_{mj}^k) \in \mathcal{Q}_k^m$ for $j \in J_N$. The per-vector average distortion D_k associated with this quantization operation can be expressed as,

$$D_k = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{m} d(x_{ij}, y_{ij}^k) = \frac{1}{N}\sum_{l=1}^{n} \sum_{\substack{i,j:y_{ij}^k = q_l^k \\ i \in J_m \\ j \in J_N}} d(x_{ij}, q_l^k). \tag{4}$$

2. Update the set of quantization levels to $\mathcal{Q}_{k+1} = \{q_1^{k+1}, q_2^{k+1}, \ldots, q_n^{k+1}\}$ where

$$q_l^{k+1} = \underset{q}{\operatorname{argmin}} \sum_{\substack{i,j:y_{ij}^k = q_l^k \\ i \in J_m \\ j \in J_N}} d(x_{ij}, q), \quad l \in J_n. \tag{5.a}$$

It is straightforward to show that $\mathcal{Q}_{k+1}$ minimizes D_k and hence reduces distortion over $\mathcal{Q}_k$. In the important special case of squared-error distortion ($d(x,y) = (x-y)^2$),

[5] The procedure for choosing $\mathcal{Q}_0$ for the first visit to Step A will be described later.

it is easily shown that (5.a) results in

$$q_l^{k+1} = \left[\sum_{\substack{i,j:y_{ij}^k = q_l^k \\ i \in J_m \\ j \in J_N}} 1 \right]^{-1} \left[\sum_{\substack{i,j:y_{ij}^k = q_l^k \\ i \in J_m \\ j \in J_N}} x_{ij} \right], \quad l \in J_n. \tag{5.b}$$

3. Set $k = k+1$ and quantize each vector in $\mathcal{X}$ using the SVQ defined by $(\mathcal{Q}_k, \mathcal{L}, L)$ and compute the corresponding average distortion D_k. If $(D_{k-1} - D_k)/D_{k-1} \leq \epsilon$, stop with $\mathcal{Q} = \mathcal{Q}_k$; else go to Step 2.

Since the updating in Step 2 above can only reduce the distortion, the above algorithm converges for a given $(\mathcal{L}, L)$ resulting in $\mathcal{Q}_{opt}(\mathcal{L}, L)$.

B. Step B: Determination of $\mathcal{L}$ and L given $\mathcal{Q}$

As mentioned before, the optimal solution for $\mathcal{L}$ and L given $\mathcal{Q}$ is not known. Two different suboptimal solutions for determining $\mathcal{L}$ are proposed here. The first solution (Method 1) is based upon choosing the 2^{mr} most probable vectors in $\mathcal{Q}^m$ as the codevectors; the second approach (Method 2) is based on the asymptotic level probabilities for large values of m. After obtaining $\mathcal{L}$ using either one of these methods, L is obtained as the maximum value of the threshold for which there are at most 2^{mr} vectors in the codebook. An algorithm to do this is presented in Section III.*B.3*.

B.1. Determination of $\mathcal{L}$ given $\mathcal{Q}$: Method 1

The m-dimensional grid $\mathcal{Q}^m$ contains n^m points out of which only 2^{mr} points should be selected as codevectors. In this method, $\mathcal{L}$ is determined such that the SVQ codevectors correspond to the 2^{mr} *most probable* grid-points. Clearly this choice does not necessarily minimize the average distortion but it ensures that with the highest probability the input vector will be quantized to the minimum distortion point on the grid.

Let p_i, $i \in J_n$ be the probability that the input sample is quantized to level q_i if minimum distortion scalar quantization (quantization to the nearest level) is performed. These probabilities can either be calculated from the source *pdf* or the set of training vectors. Since the source is a stationary memoryless source, the probability $P(\mathbf{v})$ of an arbitrary grid-point $\mathbf{v} = (v_1, v_2, \ldots, v_m) \in \mathcal{Q}^m$ is given by

$$P(\mathbf{v}) = \prod_{l=1}^{m} p_{f(v_l)} \quad = 2^{-\sum_{l=1}^{m} \log_2(1/p_{f(v_l)})}, \tag{6}$$

where f is the index function defined in (2). If we define the length ℓ_i as,

$$\ell_i = \log_2(1/p_i),\ i \in J_n, \tag{7}$$

we have,

$$P(\mathbf{v}) = 2^{-\sum_{l=1}^{m} \ell_{f(v_l)}}, \tag{8}$$

implying that the higher probability grid-points are indeed the ones with smaller total lengths. However there is a problem — the lengths obtained using (7) are not integers. This can be overcome by modifying (7) to

$$\ell_i = [\log(1/p_i)],\ i \in J_n, \tag{9.a}$$

where the square brackets denote rounding off to the nearest integer. But this could lead to a large round-off error in which case a smaller total length will not necessarily imply higher probability. Since the SVQ is essentially unchanged if all the lengths in $\mathcal{L}$ and the threshold L are multiplied by the same number, the effective round-off error of (9.a) could be reduced by defining ℓ_i as,

$$\ell_i = [b\log(1/p_i)],\ i \in J_n, \tag{9.b}$$

for some $b \in \mathbb{R}$, where larger values of b correspond to smaller effective round-off errors. Although increasing the value of b results in a better correspondence between smaller total length and higher probability, it also increases the complexity of implementation of the SVQ (Sections IV and VI). The effect of the choice of b on the performance of the SVQ is discussed in Section VII.

Although the above suboptimal scheme is expected to perform well because it maximizes the probability of quantization to the nearest grid-point, it has one drawback — it is not necessarily asymptotically (in block-length m) optimal. In other words, the performance of the SVQ designed using this scheme does not necessarily approach the performance of the optimal ECSQ as m increases. We now present another scheme for determining $\mathcal{L}$ given $\mathcal{Q}$ which is motivated by the design algorithm of the ECSQ and is expected to lead to good designs even for large values of m.

B.2. Determination of $\mathcal{L}$ given $\mathcal{Q}$: Method 2

This method is motivated by the design of the ECSQ [2]. For the given $\mathcal{Q}$, we first determine the set of quantization thresholds $\mathcal{T} \equiv \{t_0, t_1, \ldots, t_n\}$ for a minimum distortion scalar quantizer such that its output entropy is no greater than the design rate r. The probabilities of the quantization regions (intervals) of this quantizer are then used in (9.b) to determine the lengths. Thus, we need to determine $\mathcal{T}$ to minimize the distortion

$$D(\mathcal{T}) = \sum_{i=1}^{n} \int_{t_{i-1}}^{t_i} d(x, q_i) p(x) dx, \tag{10.a}$$

subject to the constraint on the output entropy

$$H(\mathcal{T}) = -\sum_{i=1}^{n} p_i \log_2 p_i \leq r, \tag{10.b}$$

where $p_i = \int_{t_{i-1}}^{t_i} p(x)dx$ is the probability[6] of quantizing to the i^{th} level when scalar quantization is performed. Writing the Kuhn-Tucker [22] necessary conditions for the locally optimal solution $\mathcal{T}^*$, we get that $\exists \lambda \in \mathbb{R}$ and $\lambda \geq 0$ such that

$$\nabla D(\mathcal{T}^*) + \lambda \nabla H(\mathcal{T}^*) = 0, \tag{11.a}$$

and

$$\lambda[H(\mathcal{T}^*) - r] = 0\ . \tag{11.b}$$

For the special case of $d(x, y) = (x - y)^2$, (11) reduces to [2]

$$\lambda \log(p_{i+1}^*/p_i^*) = (q_{i+1} - q_i)(q_{i+1} + q_i - 2t_i^*),\ i \in J_{n-1}, \tag{12}$$

in which the p_i^*'s are the level probabilities corresponding to $\mathcal{T}^*$. Rearranging (12) we get

$$\begin{aligned} t_i^* &= \frac{q_{i+1} + q_i}{2} - \frac{\lambda}{2(q_{i+1} - q_i)} \log(p_{i+1}^*/p_i^*) \\ &\triangleq u_i(\mathcal{T}^*), \quad \text{for} \quad i \in J_{n-1}. \end{aligned} \tag{13}$$

[6] If the source *pdf* is not known and the SVQ design is based on a training sequence approach, the integrals in this equation can be replaced by the corresponding sum over the training sequence.

In the vector form this can be written as,

$$\mathcal{T}^* = [u_1(\mathcal{T}^*), u_2(\mathcal{T}^*), \ldots, u_{n-1}(\mathcal{T}^*)]^T = \mathbf{u}(\mathcal{T}^*). \tag{14}$$

Hence, the solution $\mathcal{T}^*$ is a fixed point of the mapping $\mathbf{u}(\cdot)$. This suggests the following procedure for determining $\mathcal{T}$ which is similar to Algorithm 2 in [2].

0. Choose a small $\Delta\lambda > 0$; set $\lambda = 0$.
1. Determine the fixed point $\mathcal{T}^*$ of the mapping $\mathbf{u}(\cdot)$ in (14).
2. If $H(\mathcal{T}^*) > r$, then set $\lambda = \lambda + \Delta\lambda$ and go to Step 1; else stop with $\mathcal{T}^*$ as the solution.

After determining the required $\mathcal{T}$ and the corresponding p_i's, the lengths ℓ_i, for $i \in J_n$ are evaluated using (9.b). Although we do not prove it here, it is expected that (for large values of b) the performance of the SVQ designed using this scheme will approach the performance of the (locally) optimal ECSQ as block-length increases.

B.3. Determination of the threshold L given $\mathcal{L}$

For a given $\mathcal{L}$ the threshold L is obtained by counting the grid-points (starting with the ones that have the smallest total length) until there are 2^{mr} points and then taking the largest total length for which all grid-points of that length are included in this collection.

Let N_i^j represent the number of distinct i-vectors $(v_1, v_2, \ldots, v_i) \in \mathcal{Q}^i$ such that their total length $\ell_{f(v_1)} + \ell_{f(v_2)} + \cdots + \ell_{f(v_i)} = j$. Then N_i^j satisfies the following recursive relationship $\forall\, i \in J_m$:

$$N_i^j = \sum_{k=1}^{n} N_{i-1}^{j-\ell_k} \ , \tag{15}$$

where $N_i^j = 0$ for $j < 0$ and $N_0^0 = 1$. The N_i^j's can hence be determined by solving these equations. The total number C_m^j of grid-points with total length less than or equal to j is now given as,

$$C_m^j = \sum_{k=1}^{j} N_m^k \ . \tag{16}$$

The threshold L can therefore be evaluated as,

$$L = \max\{j : \ C_m^j \leq 2^{mr}\} \ . \tag{17}$$

This choice of L guarantees that there will be at most 2^{mr} vectors in the codebook and hence each codevector can be encoded by an mr-bit binary codeword.

C. SVQ design algorithm

Having described Steps A and B, we can now describe the overall algorithm used for SVQ design. As mentioned before, the algorithm consists of an iterative application of Steps A and B. Since the optimal solution for Step B is not found, the convergence of the algorithm is not guaranteed. Therefore, for stopping criterion we use a combination of a threshold on the relative distortion reduction in two successive iterations and a maximum and minimum limit on the number of iterations. Design algorithms using Methods 1 and 2 for Step B are referred to as Algorithms 1 and 2, respectively.

Algorithm 1(2)

0. Let $\epsilon > 0$, $k_{\max}$ and $k_{\min}$ denote the stopping threshold, the maximum and the minimum number of iterations. For a given training sequence $\mathcal{X}$ and a fixed number of quantization levels n, design an initial scalar quantizer with quantization levels[7] $\mathcal{Q}_0$; for $\mathcal{Q}_0$, determine from Step B, the set of lengths $\mathcal{L}_0$ and the threshold L_0; set the iteration index $k = 0$; set initial average distortion $D_0 = \infty$.
1. For fixed $\mathcal{L}_k$ and L_k, use Step A (with $\mathcal{Q}_k$ as initial set of quantization levels) to obtain $\mathcal{Q}_{k+1} \equiv \mathcal{Q}_{opt}(\mathcal{L}_k, L_k)$. Compute the average distortion D_{k+1} associated with the triple $(\mathcal{Q}_{k+1}, \mathcal{L}_k, L_k)$. If $(D_k - D_{k+1})/D_k < \epsilon$ and $k \geq k_{\min}$, stop with $(\mathcal{Q}_{k+1}, \mathcal{L}_k, L_k)$ defining the SVQ, else if $k = k_{\max}$, stop and select the SVQ as that which resulted in the smallest average distortion among all iteration up to $k_{\max}$; else, set $k = k + 1$.
2. For $\mathcal{Q}_k$, determine from Step B, the set of lengths $\mathcal{L}_k$ and the threshold L_k. Go to Step 1.

IV. Codebook Search and Encoding of the SVQ

The implementation of the SVQ defined by $(\mathcal{Q}, \mathcal{L}, L)$ essentially involves two steps: (i) the codebook search (finding the vector in the codebook closest to the input vector) and (ii) the encoding of the codevector (determining the binary codeword corresponding to the codevector). Algorithms that exploit the codebook structure for a fast and efficient implementation of both these steps are presented below.

[7] In general the SVQ design algorithm is sensitive to the choice of $\mathcal{Q}_0$. For SVQs designed in Section VII we started with uniform quantizers with appropriately chosen stepsize to ensure the rate constraint.

A. Codebook search

The aim here is to find the vector in $\mathcal{Q}^m$ of total length less than or equal to L (hence in $\mathcal{Z}$) that is closest (in the sense of minimum distortion) to the given input vector. This can be done efficiently by a dynamic programming [23] algorithm similar to the Viterbi algorithm for convolutional coding [24].

Let D_i^j be the minimum distortion that results when the first i components of the input m-vector $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ are quantized to any of the i-vectors $\mathbf{z}_i = (z_1, z_2, \ldots, z_i) \in \mathcal{Q}^i$ such that the total length of $\mathbf{z}_i$ is j, i.e., $\ell_{f(z_1)} + \ell_{f(z_2)} + \cdots + \ell_{f(z_i)} = j$. Also, let $\mathbf{z}_i^j$ represent the i-vector that results in the minimum distortion D_i^j. The distortion D_{i+1}^j is then recursively given as,

$$D_{i+1}^j = \min_{k \in J_n} \left[D_i^{j-\ell_k} + d(x_{i+1}, q_k) \right], \; i = 0, 1, \ldots, m-1, \tag{18}$$

where $D_i^j = \infty$ for $j \leq 0$ or $i \leq 0$ but $D_0^0 = 0$.

If the above is minimized for $k = k'$, then the corresponding minimum distortion vector $\mathbf{z}_{i+1}^j$ is given by the recursive equation,

$$\mathbf{z}_{i+1}^j = (\mathbf{z}_i^{j-\ell_{k'}}, q_{k'}) \; . \tag{19}$$

Using the dynamic programming algorithm to solve these equations we can determine D_m^j, $\forall j \in J_L$. The minimum distortion $D_{\min}$ (for the codebook search) is then given by

$$D_{\min} = \min_{j \in J_L} \; D_m^j \; . \tag{20}$$

If the minimization in (20) is achieved with $j = j'$, then $\mathbf{z}_m^{j'}$ gives the desired codevector.

B. Encoding of codevectors

There are 2^{mr} codevectors in the codebook $\mathcal{Z}$ of the SVQ. The encoder is a mapping which assigns mr-bit binary codewords (or simply an index corresponding to the integer representation of the codeword) to the codevectors in a one-to-one manner. The following algorithm presents one such mapping.

Every codevector $\mathbf{z} = (z_1, z_2, \ldots, z_m) \in \mathcal{Z}$ can be represented by an m-digit base-n number $\mathcal{N}(\mathbf{z})$ given as,

$$\begin{aligned} \mathcal{N}(\mathbf{z}) &= (f(z_1) - 1, \; f(z_2) - 1, \; \ldots, \; f(z_m) - 1) \\ &= \sum_{l=1}^{m} n^{m-l} (f(z_l) - 1). \end{aligned} \tag{21}$$

Clearly, $\mathcal{N}(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_2) \Leftrightarrow \mathbf{z}_1 = \mathbf{z}_2$. The encoder function $\gamma : \mathcal{Z} \rightarrow \{0\} \cup J_{2^{mr}-1}$ is then defined as,

$$\gamma(\mathbf{z}) = \sum_{\mathbf{w} \in \mathcal{Z}} I_{\mathbf{z},\mathbf{w}} \ , \tag{22.a}$$

where

$$I_{\mathbf{z},\mathbf{w}} = \begin{cases} 0, & \text{if } \mathcal{N}(\mathbf{w}) \geq \mathcal{N}(\mathbf{z}) \\ 1, & \text{if } \mathcal{N}(\mathbf{w}) < \mathcal{N}(\mathbf{z}). \end{cases} \tag{22.b}$$

In other words, $\gamma(\mathbf{z})$ is the number of codevectors in $\mathcal{Z}$ that are "smaller" than $\mathbf{z}$ (smaller in the sense $\mathcal{N}(\mathbf{w}) < \mathcal{N}(\mathbf{z})$). We can also write $\gamma(\mathbf{z})$ as,

$$\gamma(\mathbf{z}) = \sum_{k=1}^{m} \gamma_k(\mathbf{z}) \ , \tag{23}$$

where $\gamma_k(\mathbf{z})$ is the number of codevectors $\mathbf{w} \in \mathcal{Z}$ such that $\mathcal{N}(\mathbf{w})$ and $\mathcal{N}(\mathbf{z})$ represented as base-n numbers are equal in their $k-1$ most significant digits while the k^{th} digit of $\mathcal{N}(\mathbf{w})$ is smaller than that of $\mathcal{N}(\mathbf{z})$.

Let C_i^j represent the number of distinct i-vectors $(z_1, z_2, \ldots, z_i) \in \mathcal{Q}^i$ such that $\ell_{f(z_1)} + \ell_{f(z_2)} + \cdots + \ell_{f(z_i)} \leq j$. Also, define

$$C_0^j = \begin{cases} 0, & \text{if } j < 0 \\ 1, & \text{if } j \geq 0 \ . \end{cases} \tag{24}$$

Then, it can be shown that

$$\gamma_k(\mathbf{z}) = \sum_{j=1}^{f(z_k)-1} C_{m-k}^{L-L_{k-1}-\ell_j}, \quad k \in J_m, \tag{25}$$

where $L_0 = 0$ and $L_i = \sum_{j=1}^{i} \ell_{f(z_j)}$, $i \in J_m$. Combining (23) and (25) yields the following explicit expression for the index of the codeword associated with each codevector $\mathbf{z}$:

$$\gamma(\mathbf{z}) = \sum_{k=1}^{m} \sum_{j=1}^{f(z_k)-1} C_{m-k}^{L-L_{k-1}-\ell_j} \ . \tag{26}$$

Note that C_i^j can be expressed in terms of the N_i^j's (Section III) as,

$$C_i^j = \sum_{k=1}^{j} N_i^k, \quad \forall i \in J_m \text{ and } j > 0 \,. \tag{27}$$

For fast encoding, the C_i^j's can be evaluated once and stored in the memory.

V. Performance Bounds

In this section bounds on the performance of the SVQ are established and its performance is compared with that of the LMQ, the ECSQ and the permutation encoder. An m-dimensional SVQ derived from an n-level SQ (hereafter called SVQ with n scalar levels) is referred to as *optimal* if it results in a smaller average distortion than any other m-dimensional SVQ with n scalar levels at the same rate. It can be shown that optimal SVQs with n scalar levels exist for a large class of sources. This is because there are only a finite number of ways one can choose 2^{mr} codevectors from n^m grid points. Hence, for every set $\mathcal{L}$ and a corresponding threshold L that result in a distinct codebook we can determine the optimal $\mathcal{Q}$ using Step A of Section III.*A*.

We begin by showing that the performance of the SVQ is upper-bounded by the performance of the optimal ECSQ. This is done with the help of Theorem 1 and Conjecture 1. Next, we show in Theorem 2 that the performance of the optimal SVQ can approach that of the ECSQ as the block-length increases. These two bounds together imply that for a sufficiently large block-length the optimal SVQ is equivalent to the optimal ECSQ in its rate-distortion performance.

Theorem 1: Given an $\epsilon > 0$, $\exists$ an integer $M > 0$ such that the expected distortion[8] of an m-dimensional, r bit/sample SVQ with n scalar levels is no less than the expected distortion of an optimal n-level ECSQ of rate (entropy) less than $r + \epsilon$ whenever $m > M$.

Proof: Let the optimal SVQ $\mathcal{V}$ of block-length m have a set of quantization levels $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$, a corresponding set of lengths $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_n\}$ and a threshold total length L. Note that $\mathcal{Q}$, $\mathcal{L}$ and L are functions of m. Let $\mathcal{Z}$ denote the codebook of $\mathcal{V}$.

If $\mathbf{p} \equiv (p_1, p_2, \ldots, p_n)$ is the type [25] of the codevector $\mathbf{z} \in \mathcal{Z}$, then $p_i m \in \{0\} \cup J_m \; \forall \, i \in J_n$. Let $\mathcal{P}_m$ be the set of all the possible types of codevectors, i.e.,

$$\mathcal{P}_m \equiv \{\mathbf{p} : \quad p_1 m\ell_1 + p_2 m\ell_2 + \cdots + p_n m\ell_n \leq L\}. \tag{28}$$

[8] The distortion $d(x, y)$ here is assumed to be of the form, $d(x, y) = |x - y|^\beta$ where $\beta \geq 1$.

The total number of codevectors N in $\mathcal{Z}$ is then given as,

$$N = \sum_{\mathbf{p} \in \mathcal{P}_m} \frac{m!}{\prod_{i=1}^{n} (mp_i)!} \,. \tag{29}$$

Using the Stirling's inequalities (for integer $i > 0$) [26],

$$\sqrt{2\pi}\, i^{(i+1/2)} e^{-i} < i! < \sqrt{2\pi}\, i^{(i+1/2)} e^{-i} \left(1 + \frac{1}{12i-1}\right), \tag{30}$$

it is straightforward to show that

$$N > \sum_{\mathbf{p} \in \mathcal{P}_m} O_m 2^{mh(\mathbf{p})} \,, \tag{31}$$

where O_m denotes the fraction of two polynomials in $\sqrt{m}$ and $h(\mathbf{p})$ is the entropy associated with the type $\mathbf{p}$. If $\mathbf{p}_{\max} \in \mathcal{P}_m$ is the type that has the maximum entropy, it follows from (31) that

$$N > O_m \; 2^{mh(\mathbf{p}_{\max})}. \tag{32}$$

The rate r of $\mathcal{V}$, therefore, satisfies

$$r = \frac{1}{m} \log_2 N > \frac{\log_2 O_m}{m} + h(\mathbf{p}_{\max}) \,. \tag{33}$$

Each type corresponds to a point in an n-dimensional space. The set $\mathcal{P}_m$ is a subset of the closed, convex and bounded set $\mathcal{A}_m$ described by,

$$\mathcal{A}_m = \{\mathbf{a} \equiv (a_1, a_2, \ldots, a_n) : a_1 m\ell_1 + a_2 m\ell_2 + \cdots + a_n m\ell_n \leq L,$$

$$a_i \in \mathbb{R},\; a_i \geq 0 \;\forall\; i \in J_n \;\text{ and }\; \sum_{i=1}^{n} a_i = 1\} \,. \tag{34}$$

It is easy to see that given any $\dot{\delta} > 0$ there exists an integer I such that for all $\mathbf{a} \in \mathcal{A}_m$, $\max\limits_{i \in J_n} |a_i - p_i| < \delta$ for some $\mathbf{p} \in \mathcal{P}_m$ whenever $m > I$. In other words as m becomes large, every point of $\mathcal{A}_m$ can be approximated by a type in $\mathcal{P}_m$. Also, since entropy is a continuous function on a closed and bounded set $\mathcal{A}_m$, it is uniformly continuous and the following can easily be concluded

$$\max_{\mathbf{a} \in \mathcal{A}_m} h(\mathbf{a}) = \max_{\mathbf{p} \in \mathcal{P}_m} h(\mathbf{p}) + \delta_1(m) = h(\mathbf{p}_{\max}) + \delta_1(m) < r + \delta(m), \tag{35}$$

where $\delta(m)$ and $\delta_1(m)$ are non-negative terms going to zero as $m \to \infty$.

Now consider a vector quantizer $\mathcal{V}_k$ of block-length km such that its codebook $\mathcal{Z}_k$ consists of all possible concatenations of k codevectors of $\mathcal{V}$ as well as every permutation of each of its codevectors. Clearly, for any input, the distortion of $\mathcal{V}_k$ will be no greater than that of $\mathcal{V}$. Next, it is shown that as k increases, the expected distortion of $\mathcal{V}_k$ approaches that of a scalar quantizer $\mathcal{S}_m$ with rate (entropy) not exceeding $r + \delta(m)$.

Let $\mathcal{P}_m^k$ be the set of all types of codewords in $\mathcal{Z}_k$. Then it is obvious from the construction of $\mathcal{Z}_k$ that

$$\mathbf{p}^k \in \mathcal{P}_m^k \Rightarrow \mathbf{p}^k = (i_1/k)\mathbf{p}_1 + (i_2/k)\mathbf{p}_2 + \cdots + (i_k/k)\mathbf{p}_k, \tag{36}$$

where $i_1, i_2, \ldots, i_k$ are non-negative integers with $i_1 + i_2 + \cdots + i_k = k$ and $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k \in \mathcal{P}_m$. This shows that $\mathcal{P}_m^k \subseteq co(\mathcal{P}_m)$, where $co(\cdot)$ denotes the convex hull. Also, as k becomes large, any point in the closure of the convex hull of $\mathcal{P}_m$ (denoted by $\overline{co(\mathcal{P}_m)}$) can be well approximated by a type in $\mathcal{P}_m^k$. Since $\mathcal{P}_m \subseteq \mathcal{A}_m$ which is convex, we conclude that $\mathcal{P}_m^k \subseteq \mathcal{A}_m$. If $\mathcal{T}_{\mathbf{p}^k}$ is the set of those codevectors of $\mathcal{Z}_k$ which have a type $\mathbf{p}^k$, then clearly $\mathcal{T}_{\mathbf{p}^k}$ defines the codebook of a permutation coder $\mathcal{E}_{\mathbf{p}^k}$ and we have,

$$\mathcal{Z}_k = \bigcup_{\mathbf{p}^k \in \mathcal{P}_m^k} \mathcal{T}_{\mathbf{p}^k}. \tag{37}$$

Let $\mathbf{p}' = (p_1', p_2', \ldots, p_n')$ be a point in $\overline{co(\mathcal{P}_m)}$. Consider a scalar quantizer $\mathcal{S}_{\mathbf{p}'}$ with quantization levels $\mathcal{Q}$ and the quantization thresholds $\{t_0, t_1, \ldots, t_n\}$ chosen such that the distortion is minimized while the probability of the quantization region of q_i is p_i' for $i \in J_n$. Berger has shown (proof of Theorem 2 in [15]) that, for the distortion measure of footnote 8, a j-dimensional permutation coder $\mathcal{E}_{\mathbf{p}^j}$ with quantization levels $\mathcal{Q}$ and type $\mathbf{p}^j = (p_1^j, p_2^j, \ldots, p_n^j)$ becomes equivalent to $\mathcal{S}_{\mathbf{p}'}$ as $j \to \infty$ provided that $\lim_{j\to\infty} p_i^j = p_i' \ \forall i \in J_n$. The equivalence is in the sense that the rate of the permutation encoder converges to the rate (entropy) $h(\mathbf{p}')$ of the quantizer while the sample-average distortion $d(\mathcal{E}_{\mathbf{p}^j})$ of the permutation encoder converges to the expected distortion $\mathrm{E}(d(\mathcal{S}_{\mathbf{p}'}))$ of the quantizer with probability one.

For $\mathbf{p}' \in \overline{co(\mathcal{P}_m)}$, let $\mathbf{p}^k = \mathbf{p}''$ be the type in $\mathcal{P}_m^k$ that is closest (in l^∞ norm) to $\mathbf{p}'$. Clearly, $\lim_{k\to\infty} p_i'' = p_i' \ \forall i \in J_n$. This is because every point in $\overline{co(\mathcal{P}_m)}$ is well approximated by a type in $\mathcal{P}_m^k$ for a large enough k. Hence, by Berger's Theorem [15],

$$d(\mathcal{E}_{\mathbf{p}''}) \to \mathrm{E}(d(\mathcal{S}_{\mathbf{p}'})) \ \text{ a.s. } \ \text{as} \ \ k \to \infty. \tag{38}$$

Now the codebook $\mathcal{Z}_k$ given by (37) can also be represented as,

$$\mathcal{Z}_k = \bigcup_{\mathbf{p}' \in \overline{co(\mathcal{P}_m)}} \mathcal{T}_{\mathbf{p}''}, \tag{39}$$

where $\mathbf{p}''$ as before is the type in $\mathcal{P}_m^k$ that is closest to $\mathbf{p}'$. Hence, the sample-average distortion $d(\mathcal{V}^k)$ of the quantizer $\mathcal{V}^k$ as $k \to \infty$ is given as,

$$\lim_{k\to\infty} d(\mathcal{V}^k) = \lim_{k\to\infty} \inf_{\mathbf{p}' \in \overline{co(\mathcal{P}_m)}} d(\mathcal{E}_{\mathbf{p}''}). \tag{40}$$

It can be argued [27] that the limit and the infimum can be interchanged in the previous equation giving

$$\lim_{k\to\infty} d(\mathcal{V}^k) = \inf_{\mathbf{p}' \in \overline{co(\mathcal{P}_m)}} \mathrm{E}(d(\mathcal{S}_{\mathbf{p}'})) \quad \text{a.s.} . \tag{41}$$

Let $\mathbf{p}' = \mathbf{p}_{\min} \in \overline{co(\mathcal{P}_m)}$ be the distribution for which the corresponding scalar quantizer $\mathcal{S}_{\mathbf{p}'} = \mathcal{S}_m$ has minimum expected distortion $d_{\min}$. Now, as shown in [15], for all sources for which $\mathrm{E}|X_n|^\beta < \infty$ we have,

$$\mathrm{E}(d(\mathcal{V}_k)) \to d_{\min} \quad \text{as} \quad k \to \infty. \tag{42}$$

The rate R of $\mathcal{S}_m$ is given as,

$$R = h(\mathbf{p}_{\min}). \tag{43}$$

But $\mathbf{p}_{\min} \in \overline{co(\mathcal{P}_m)} \subseteq \mathcal{A}_m$. This shows that (using (35))

$$R = h(\mathbf{p}_{\min}) \leq \max_{\mathbf{a} \in \mathcal{A}_m} h(\mathbf{a}) < r + \delta(m). \tag{44}$$

Therefore there exists an entropy coded scalar quantizer $\mathcal{S}_m$ that has an expected distortion no greater than that of $\mathcal{V}$ (using (42)) and a rate no greater than $r + \delta(m)$. Since $\delta(m) \to 0$ as $m \to \infty$, we can find an integer M such that $\delta(m) < \epsilon$ for $m > M$. □

This theorem bounds the performance of large block-length SVQs. We next examine the SVQ performance as a function of its block-length. Since with increasing block-length the observed distribution of the source samples in the block comes closer to the true probability distribution of the source, it is intuitively satisfying to assume that the performance

of the optimal SVQ (designed on the source probability distribution) improves as the block-length increases. This is reflected in the following conjecture.

Conjecture 1: For a given average encoding rate per sample and a fixed number of quantization levels, the performance of the optimal m-dimensional SVQ for an *iid* source improves as the block-length m increases.

Though the conjecture is stated for an optimal SVQ, it can also be expected to hold for SVQs designed using good suboptimal schemes. This was indeed the case for most SVQs designed using Algorithms 1 and 2 of Section III.

Theorem 1 together with Conjecture 1 suggests that the performance of the SVQ at any block-length is bounded by the performance of the optimal ECSQ at the same rate and the same number of quantization levels.

Next, we show that for large block-lengths the performance of the optimal SVQ indeed approaches that of the optimal ECSQ.

Theorem 2: The expected distortion of the optimal m-dimensional rate r SVQ with n scalar levels approaches that of the optimal n-level ECSQ at the same rate as the block length increases.

Proof: The theorem is proved by using the optimal ECSQ to construct an m-dimensional SVQ the average distortion and the rate of which approach the average distortion and the rate of the ECSQ as m increases. The SVQ is constructed as follows. The set of quantization levels $\mathcal{Q}$ is the same as the set of quantization levels of the optimal ECSQ. The length ℓ_i of level q_i is assigned as,

$$\ell_i = [b \log 1/p_i], \quad \text{for some } b \in \mathbb{R}, \tag{45}$$

where p_i is the probability of quantizing the input sample to q_i when using the ECSQ. The threshold L is chosen such that there are at most 2^{mr} vectors in the codebook.

Assume that b is sufficiently large so that length quantization can be ignored. Of all the different vector types in the codebook of the SVQ, let $\mathbf{p}^m$ be the one that is closest to $(p_1, p_2, \ldots, p_n)$. It is now easy to show that as m increases, vectors of type $\mathbf{p}^m$ dominate the rate and distortion of the SVQ and the SVQ becomes equivalent to a permutation coder of type $\mathbf{p}^m$. But for large m, $\mathbf{p}^m$ approaches $(p_1, p_2, \ldots, p_n)$ and by Theorem 2 in [15] the permutation coder is equivalent to the ECSQ. □

An LMQ is a special case of the optimal SVQ when the block-length is one. Hence by Conjecture 1, the optimal SVQ performs better than the LMQ. The results in Section VII indicate that this conclusion is generally expected to hold for suboptimal SVQs of Section III.

For relatively small block-lengths the permutation encoder performs poorly [15] and as supported by the results in Section VII, the SVQ can be expected to perform better. For large block-lengths the performance of both of these approaches that of the ECSQ.

VI. Complexity Issues

In this section we analyze the complexity of implementation of the SVQ for the squared-error distortion measure; the analysis can easily be extended to other distortion measures. We will consider both computational and storage complexities associated with the codebook search and codevector encoding algorithms. The computational complexity is determined in terms of the total number of operations (multiplications, additions or comparisons) required *per source sample* while the storage complexity is measured as the *total* number of 32-bit memory words required by the algorithm.

A. Computational complexity

First, consider the codebook search algorithm. This corresponds to the determination of the codevector that achieves the minimum distortion. Recall from Subsection IV.*A* that the distortion incurred in the encoding operation is computed based on equation (18), or equivalently, the following more efficient equation:

$$D_{i+1}^{j} = \min_{\ell \in \mathcal{L}} [D_i^{j-\ell} + d_\ell(x_{i+1})], \; i = 0, 1, \ldots, m-1, \tag{46}$$

where $d_\ell(x)$ is the minimum distortion that results when x is quantized to all those levels that have length ℓ. For each input vector, these distortions can be computed once and stored. If the number of distinct length values in $\mathcal{L}$ is denoted by n', determining these distortions (for the squared-error distortion measure) requires $(3n - n')m$ operations (n additions, n multiplications and $n - n'$ comparisons for each component). The number of operations for determining each D_i^j according to (46) is $2n' - 1$ which corresponds to a total of $(2n' - 1)mL$ operations for determining the D_i^j's and hence the minimum distortion. This adds up to a total of $(2n' - 1)mL + (3n - n')m$ operations for the search which corresponds to $C_{\text{srch}} = (2n' - 1)L + (3n - n')$ operations per source sample.

Based on the SVQs designed thus far, the following observations can be made: (i) the required number of quantization levels n for good performance is exponential in the per sample rate r (see also [2]), (ii) for the SVQs designed using the algorithms of Section III the threshold L is of the order of mrb (for all cases considered $L < 2mrb$; later in this section we describe ways to reduce L without changing the codebook of the SVQ) and (iii) while in general $n' \leq n$, for high rates, $n' \ll n$. In addition, it is easy to see that $n' < L$. These observations imply that the SVQ search complexity is at most exponential in r and polynomial in m. This is simpler than the codebook search in a full-search VQ where the complexity is exponential in mr. Finally, note the direct dependence of this complexity on the parameter b.

As for the encoding operation of Subsection IV.B, we need to use (26) to determine $\gamma(\mathbf{z})$. The inner sum in (26) is performed at most $n-1$ times while the outer sum is performed $m-1$ times. So assuming that the C_i^j's are available (stored), at most $(m-1)(n-1)$ additions are required for this double summation. This corresponds to approximately n additions per source sample. Note however that C_i^j's are integers and could have large binary representations (up to mr bits). Hence if each 32 bit addition represents an operation, $C_{\text{enc}} \leq nmr/32$ operations are required per source sample. The encoding complexity is usually small compared to that of the codebook search.

Remark:

It is interesting to note that the codebook of the SVQ remains unaltered if all the lengths in $\mathcal{L}$ are increased (decreased) by an integer l and the threshold is increased (decreases) by ml. Similarly, the codebook is also unchanged if all lengths in $\mathcal{L}$ and the threshold are multiplied or divided (if this results in integer values of lengths) by the same integer. For an SVQ described by $(\mathcal{Q}, \mathcal{L}, L)$ the above two observations can be used to modify $\mathcal{L}$ such that the value of the threshold L, and hence the search complexity, are reduced while keeping the quantizer unchanged. As an example, consider a 4-dimensional SVQ $\mathcal{V}$ with $\mathcal{L} = \{6, 13, 20\}$ and $L = 45$. We can construct an equivalent SVQ $\mathcal{V}'$ by first adding 1 to all lengths and adding 4 to the threshold and then dividing everything by 7 to get $\mathcal{L}' = \{1, 2, 3\}$ and threshold $L' = 7$. Obviously the complexity of $\mathcal{V}'$ is smaller than that of $\mathcal{V}$.

B. Storage complexity

For the codebook search, solving the recursive equation (46) requires an array of size

sources with squared-error as the measure of distortion. As mentioned before, because of the suboptimality of Step B of these design algorithms, their convergence is not guaranteed. For many quantizers designed using these algorithms, the solution converged. For others it did not converge but the average distortion initially decreased and then oscillated in a narrow range. In the latter case we picked the quantizer that resulted in the least average distortion (up to 15 iterations). For those cases where the solution converged, the convergence does not imply any optimality or even local optimality; the solution can converge because the lengths in $\mathcal{L}$ take only discrete values (integers) and hence a small change in $\mathcal{Q}$ (from Step A) could result in the same $\mathcal{L}$.

A. Rate vs. distortion characteristics of the SVQ

First, we consider the SVQs derived from ECSQ (beginning of Section III). Performance of these quantizers was evaluated based on simulations performed on 20,000 source vectors. Table 2 gives the signal-to-noise ratio (SNR) in dB for SVQs derived from ECSQ for the three different sources at various block-lengths and coding rates. The numbers in parentheses indicate the actual rate if different from the desired rate. This could happen at low rates or small block-lengths if there is no choice of the threshold L which results in a total number of codevectors close to 2^{mr}. Although these results reflect a substantial gain over Lloyd-Max quantization (Table 3), the design algorithms of Section III perform better and are considered next.

Algorithms 1 and 2 were both used to design SVQs for a training sequence of 20,000 source vectors. The number of quantization levels n was chosen such that no further performance improvement is realized by adding more levels (see Fig. 5). The lengths in $\mathcal{L}$ (Step B) were quantized to multiples of a quarter of a bit which corresponds to $b = 4.0$ in (9.b). The performances of Algorithms 1 and 2 are summarized in Tables 3 and 4, respectively. The SVQ results in these tables were obtained on test data different from training data. A comparison between Tables 2, 3 and 4 reveals that: (i) at lower rates the performance of (SVQs designed using) Algorithm 1 is comparable to SVQs derived form ECSQ while at higher rates Algorithm 1 performs better (especially for smaller α), (ii) Algorithm 2 always performs better than ECSQ-based SVQs and (iii) for most cases Algorithm 2 outperforms Algorithm 1 but there is no clear trend. In practice both Algorithms 1 and 2 can be used to design the SVQ and the better result chosen.

Tables 3 and 4 also contain the corresponding results for LMQ, ECSQ, PC (Bergers

Variant I codes [16] with block-length 32) and LBG VQ. Because of the problems mentioned in Section I, the codebook size of the LBG quantizers considered here is limited to 1024 vectors (512 for $r = 1.5$ and 3 bits/sample). In each case, the chosen VQ codebook size is indicated in brackets. The corresponding block-size can be determined from the codebook size and the rate. The rate vs. distortion plots for the SVQ (Algorithm 2) are shown in Figs. 1 and 2 for GG sources with $\alpha = 0.5$ and 2, respectively. Plots for the LMQ, ECSQ and VQ are also included for comparison. It is clear that even for small block-lengths the SVQ performs better than the LMQ and bridges the gap between fixed-rate and variable-rate scalar quantizers. The improvement over LMQ offered by the SVQ is specially significant for broad-tailed distributions (smaller α). For the block-lengths considered here, the SVQ performance is much better than the PC performance (except for $\alpha = 2$ and $r = 1$ bit/sample). In most cases the SVQ also performs better than the considered LBG VQs. For example, from Table 4.c we see that an 8-dimensional SVQ at 2.5 bits/sample for a GG source with $\alpha = 0.5$ already performs 0.09 dB better than the 4-dimensional LBG VQ at the same rate while a 32-dimensional SVQ performs 2.21 dB better than the considered LBG VQ.

In view of the above observations, it can be concluded that the SVQ provides an interesting alternative for the fixed-rate quantization of broad-tailed memoryless sources, especially at high rates.

B. Effect of the block-length m

The average distortion of the SVQ (Algorithm 2) for several coding rates is plotted as a function of the block-length in Figs. 3 and 4 for GG sources with $\alpha = 0.5$ and 2, respectively. For cases where the actual rate of the SVQ was different from the desired rate, the distortion value at the desired rate was obtained by interpolation[10]. These plots show that at low rates a small block-length is adequate to achieve performance close to the optimal ECSQ performance while at higher rates larger block-lengths are required. Also the improvement in performance with block-length is more pronounced for sources with broad-tailed density (smaller α).

C. Effect of the number of quantization levels n

Fig. 5 shows the variation of the SVQ (Algorithm 1) average distortion with the

[10] The interpolation is justified because SVQ close to the desired rate can easily be obtained by including in the codebook only a subset of vectors with length equal to the threshold L.

number of quantization levels. The block-length here is 32 and the source distribution is GG with $\alpha = 1$. Plots for three different encoding rates are given. When $n = 2^r$, the SVQ reduces to m scalar LMQs. For higher values of n the performance first improves and then saturates. It was found that sources with broad-tailed density (smaller α) require a larger value of n to reach saturation. The number of levels required to saturate the performance is expected to increase exponentially with rate and this trend is somewhat visible in Fig. 5.

D. Length quantization

Though the effective quantization error in determining the lengths ℓ_i using (9.b) decreases as b increases, it is shown in Section VI that the complexity of implementation of the SVQ increases with b. This makes it important to study the dependence of the SVQ performance on b. In Fig. 6 the performance of the SVQ (Algorithm 1) is plotted as a function of b for GG sources with $\alpha = 0.5,\ 1$ and 2, $m = 16$ and $r = 2$ bits/sample. It can be seen that the performance does not depend significantly on b and does not necessarily improve as b increases. This should be expected because we use only a suboptimal algorithm for length assignment and hence small deviation in lengths (due to quantization) could degrade or improve performance. The low sensitivity of the SVQ performance on b can be exploited to reduce the complexity of implementation of the SVQ without significantly affecting performance.

E. Dependence of the threshold L on other quantizer parameters

In Section VI it was stated that the threshold L is of the order of mbr. For all SVQs designed using algorithms of Section III it was indeed found that $mbr \leq L < 2mbr$. The value of L can however be reduced by using the techniques mentioned in the remark in Section VI.

VIII. Performance in the Presence of Channel Noise

In the previous section we have studied the performance of the SVQ assuming that the channel through which the codewords are transmitted is noiseless and does not corrupt the data. Since most practical channels are not noiseless, it is also important to study the effect of channel noise on the performance of the SVQ. In fact the very idea of the SVQ was the result of an effort to improve the performance of the variable-length encoded ECSQ (VLE-ECSQ) in the presence of channel noise.

We now study the SVQ performance in the presence of channel noise and compare it with the performance of the LMQ and VLE-ECSQ. The channel is modeled as a memoryless binary symmetric channel with a known bit-error probability. Since in the presence of channel noise, the VLE-ECSQ is prone to catastrophic error propagation due to loss of synchronization, a direct comparison of the SVQ with this scheme is of little value. To make a meaningful comparison it is assumed that the output of the VLE-ECSQ is packetized using fixed-length packets. The number of samples in a packet (which is obviously not fixed) is transmitted as overhead information and protected using a rate 1/3 error correcting code and is assumed to be always decoded correctly at the receiver. With this assumption any error during transmission will affect only the samples in that packet and will not propagate across packet boundaries. Obviously, longer packets are more sensitive to channel errors but require a lower overhead information.

Results for the SVQ were obtained based on simulations involving $20,000$ error affected codewords. The SVQs were designed using Algorithm 2 with $b = 4$. For the LMQ the results were obtained analytically assuming that the quantizer outputs are coded using a Gray code. In the case of the VLE-ECSQ, the simulations involved $20,000$ error affected packets. A first-order Huffman code was used to encode the ECSQ outputs.

Figs. 7-9 show the performance of the SVQ (for various block-lengths) as a function of the channel bit-error probability for a GG source with $\alpha = 2$ at rates of 1, 2 and 3 bits/sample, respectively. The performances of the LMQ and the packetized VLE-ECSQ scheme (for three different packet sizes) at corresponding rates (including the overhead bits for the VLE-ECSQ scheme) are also plotted for comparison. Note that in Fig. 7 the performance of the packetized VLE-ECSQ scheme is not plotted because at the rate of 1 bit/sample the best *first-order* Huffman coded variable-length scheme is the two-level LMQ. These figures indicate that for a Gaussian source the best SVQ always outperforms the packetized VLE-ECSQ scheme for the entire range of channel bit-error rates considered here. The range of bit-error rates for which the SVQ performs better than the LMQ increases with coding rate. Figs. 9-11 contain similar plots at a rate of 3 bits/sample for GG sources with $\alpha = 2,\ 1$ and 0.5, respectively. These plots show that for $\alpha = 2$ and 1 the best SVQ performs better than the packetized VLE-ECSQ scheme for the entire range of error rates. For $\alpha = 0.5$ and large packet sizes (256 and 1024), the VLE-ECSQ scheme performs a little better than the SVQ for very small error rates. For these packet sizes however,

the performance of the VLE-ECSQ scheme degrades rapidly as the channel bit-error rate increases. These results and observations suggest that the SVQ is a quantization scheme which not only performs well in the absence of channel noise, but also offers reasonable robustness for operation over noisy channels.

In the absence of channel noise the performance of the SVQ improves as the block-length increases, but a larger block-length SVQ is more adversely affected by channel noise because the channel errors affect a larger number of samples. Hence for low channel bit-error rates the performance of the SVQ improves with block-length while at higher error rates the performance degrades with block-length. Between the two extremes the performance usually peaks for some value of the block-length. This behavior is reflected in Figs. 12 and 13 where the performance of the SVQ at a rate of 3 bits/sample is plotted as a function of block-length for different values of channel bit-error probability for GG sources with $\alpha = 0.5$ and 2, respectively. These plots can be used to determine the best block-lengths for a given channel.

IX. Summary and Conclusions

In this paper a fixed-rate structured vector quantization scheme for encoding stationary memoryless sources was presented. The structure of the codebook for the vector quantizer is derived from a variable-length scalar quantizer. Two suboptimal algorithms for the design of the SVQ were presented along with fast algorithms for codebook search and code-vector encoding. It was shown that the performance of the SVQ is upper-bounded by the performance of the optimal ECSQ at the same rate and that the SVQ can asymptotically (in block-length) achieve the performance of the ECSQ. The complexity of implementation of the SVQ was analyzed as a function of the encoding rate, the block-length and other quantizer parameters. Results on the performance of the SVQ for three different sources chosen from the generalized Gaussian family were presented for a variety of encoding rates and block-lengths, both in the absence and presence of channel noise.

Numerical results presented demonstrate that the SVQ effectively bridges the gap between the LMQ and the optimal ECSQ while maintaining fixed-rate outputs; the performance gains over LMQ (and even LBG VQ of comparable complexity) are significant for broad-tailed densities and at higher rates. This, along with the fact that the complexity of the SVQ is affordable even for fine quantization and relatively large block-lengths, makes the SVQ an attractive candidate for fixed-rate quantization of memoryless sources with

broad-tailed densities at high rates. Transform coefficients in discrete cosine transform coding of images [28] and subbands in subband image coding [29] are two such examples. In addition to its attractive features in the absence of channel noise, SVQ offers a reasonable level of robustness in a noisy channel environment and, in general, outperforms both LMQ and VLE-ECSQ for a useful range of channel bit-error rates.

The SVQ ideas can be extended in several directions. In [20] and [30] the extension of the SVQ to stationary independent-component vector sources (with possibly different component-distributions) is presented; this modification along with a decorrelating transformation is used to develop an extension of the SVQ for coding stationary sources with memory. A finite-state extension of the SVQ for encoding sources with memory is also considered in [20]. Another possible extension of the SVQ is to derive it from variable-length vector quantizers in place of variable-length scalar quantizers. Even for memoryless sources this could lead to performance better than the ECSQ. This idea and some other SVQ extensions are used in [31] for the efficient quantization of speech LSP parameters.

References

1. H. Gish and J. N. Pierce, "Asymptotically Efficient Quantizing," *IEEE Trans. Inform. Theory*, Vol. IT-14, pp. 676-683, Sept. 1968.
2. N. Farvardin and J. W. Modestino, "Optimum Quantizer Performance for a Class of Non-Gaussian Memoryless Sources," *IEEE Trans. Inform. Theory*, Vol. IT-30, pp. 485-497, May 1984.
3. S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. Inform. Theory*, Vol. IT-28, pp. 129-137, March 1982.
4. J. Max, "Quantizing for Minimum Distortion," *IRE Trans. Inform. Theory*, Vol. IT-6, pp. 7-12, March 1960.
5. Y. Linde, A. Buzo and R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Commun.*, Vol. COM-28, pp. 84-95, Jan. 1980.
6. R. M. Gray, "Vector Quantization," *IEEE ASSP Mag.*, Vol. 1, pp. 4-29, April 1984.
7. C. E. Shannon, "Coding Theorems for a Discrete source with a Fidelity Criterion," in *IRE Nat. Conv. Rec.*, part 4, pp. 142-163, March 1959.
8. R. M. Gray and Y. Linde, "Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources," *IEEE Trans. Commun.*, Vol. COM-30, pp. 381-389, Feb. 1982.

9. B.-H. Juang and A. H. Gray, "Multiple stage vector quantization for speech coding," *Proc. ICASSP*, pp. 597-600, April 1982.
10. M. J. Sabin and R. M. Gray, " Product Code Vector Quantizers for Waveform and Voice Coding," *IEEE Trans. Acoustics, Speech and Signal Processing,* vol. ASSP-32, pp. 474-488, June 1984.
11. N. Moayeri, D. L. Neuhoff, and W. E. Stark, "Fine-Coarse Vector Quantization," *IEEE Transactions on Signal Processing*, July 1991, to appear.
12. J. H. Conway and N. J. A. Sloane, "Fast Quantizing and Decoding Algorithms for Lattice Quantizers and Codes," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 227-232, March 1982.
13. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Kluwer Academic Press, Massachusetts, to appear, winter 1992.
14. T. Berger, F. Jelinek and J. K. Wolf, "Permutation Codes for Sources," *IEEE Trans. Inform. Theory*, Vol. IT-18, pp. 160-169, Jan. 1972.
15. T. Berger, "Optimum Quantizers and Permutation Codes," *IEEE Trans. Inform. Theory*, Vol. IT-18, pp. 759-765, Nov. 1972.
16. T. Berger, "Minimum Entropy Quantizers and Permutation Codes," *IEEE Trans. Inform. Theory*, Vol. IT-28, pp. 149-157, March 1982.
17. T. R. Fischer, "A Pyramid Vector Quantizer," *IEEE Trans. Inform. Theory*, Vol. IT-32, pp. 568-583, July 1986.
18. P. F. Swaszek and J. B. Thomas, "Multidimensional Spherical Coordinates Quantization," *IEEE Trans. Inform. Theory*, Vol. IT-29, pp. 570-576, July 1983.
19. P. F. Swaszek, "Uniform Spherical Coordinate Quantization of Spherically Symmetric Sources," *IEEE Trans. on Commun.*, Vol. COM-33, pp. 518-521, June 1985.
20. R. Laroia and N. Farvardin, "Extensions of the Fixed-Rate Structured Vector Quantizer," in preparation for submission to *IEEE Trans. Inform. Theory.*
21. T. R. Fischer, "Geometric Source Coding and Vector Quantization," *IEEE Trans. Inform. Theory*, Vol. IT-35, pp. 137-145, Jan. 1989.
22. M. S. Bazarra and C. M. Shetty, *Nonlinear Programming*, Wiley, New York, 1979.
23. R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.
24. A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, 1979.

25. I. Csiszar and J. Korner, *Information Theory*, Academic Press, Inc., New York, 1981.
26. R. E. Blahut, *Principles and Practice of Information Theory*, p. 18, Addison-Wesley Publishing Company, 1987.
27. R. Laroia, *Design and Analysis of a Structured Fixed-Rate Vector Quantizer Derived from Variable-Length Scalar Quantizers*, Ph.D Dissertation, Electrical Engineering Department, Univ. of Maryland, College Park, in preparation.
28. R. C. Reininger and J. D. Gibson, "Distributions of the Two-Dimensional DCT Coefficients of Images," *IEEE Trans. Commun.*, Vol. COM-31, pp. 835-839, June 1983.
29. N. Tanabe and N. Farvardin, "Subband Image Coding Using Entropy-Coded Quantization Over Noisy Channels," *IEEE Journal of Selected Areas in Communications*, to appear, June 1992.
30. R. Laroia and N. Farvardin, "Extension of the Fixed-Rate Structured Vector Quantizer to Vector Sources," *Proc. of the Twenty-Fifth Annual Conference on Information Sciences and Systems*, pp. 576-581, Johns Hopkins University, Baltimore, March 1991.
31 R. Laroia, N. Phamdo and N. Farvardin, "Robust and Efficient Quantization of Speech LSP Parameters Using Structured Vector Quantizers," *Proc. IEEE ICASSP*, Toronto, Canada, pp. 641-644, May 1991.

	Codebook Search	Encoding
Computational Requirement (operations per source sample)	$(2n'-1)L+(3n-n')$	$nmr/32$
Storage Requirement (32-bit words)	$(m+1)L+n'+2n+1$	$m^2Lr/32$

Table 1: Computational and Storage Requirements of the SVQ.

$\alpha = 2.0$					
Rate	Block-Length				
	4	8	16	24	32
1.0	3.44 (0.79)	4.01 (0.88)	4.39 (0.94)	4.54 (0.97)	4.64 (0.99)
1.5	6.99	7.19 (1.48)	7.41	7.50	7.55
2.0	9.56 (1.97)	10.08 (2.01)	10.29 (2.01)	10.28 (1.99)	10.38
2.5	12.39 (2.51)	12.58 (2.48)	12.94	13.11	13.15
3.0	14.87 (3.01)	15.19	15.62	15.78	15.96

Table 2.a: SNR (dB) of SVQs Derived from ECSQ for a Generalized Gaussian Source with $\alpha = 2$ at Various Rates (bits/sample) and Block-Lengths. The Numbers in Parentheses Indicate the Actual Rate if Different from the Desired Rate.

$\alpha = 1.0$					
Rate	Block-Length				
	4	8	16	24	32
1.0	3.74 (0.79)	4.70 (0.90)	5.32 (0.97)	5.57	5.54
1.5	6.78 (1.55)	7.30 (1.46)	8.02 (1.53)	8.16 (1.52)	8.23 (1.51)
2.0	8.64 (1.98)	10.04 (2.03)	10.50	10.71	10.82
2.5	10.73 (2.47)	12.19	12.96	13.32	13.45
3.0	12.41 (2.96)	14.18	15.23	15.72	15.91

Table 2.b: SNR (dB) of SVQs Derived from ECSQ for a Generalized Gaussian Source with $\alpha = 1$ at Various Rates (bits/sample) and Block-Lengths. The Numbers in Parentheses Indicate the Actual Rate if Different from the Desired Rate.

$\alpha = 0.5$					
Rate	Block-Length				
	4	8	16	24	32
1.0	5.42 (1.02)	6.62 (1.07)	7.31	7.63 (1.01)	7.86 (1.01)
1.5	6.49 (1.40)	8.80	10.08	10.54	10.75
2.0	8.69 (1.95)	11.32 (2.03)	12.32	12.86	13.06
2.5	9.65 (2.48)	13.16	14.78	15.44	15.76
3.0	11.35 (3.03)	14.38	16.06	16.80	17.11

Table 2.c: SNR (dB) of SVQs Derived from ECSQ for a Generalized Gaussian Source with $\alpha = 0.5$ at Various Rates (bits/sample) and Block-Lengths. The Numbers in Parentheses Indicate the Actual Rate if Different from the Desired Rate.

$\alpha = 2.0$									
Rate	Block-Length					LMQ	ECSQ	LBG	PC
	4	8	16	24	32			VQ	
1.0	3.42 (0.79)	4.01 (0.88)	4.43 (0.94)	4.53 (0.97)	4.61 (0.99)	4.40	4.64	5.05 [1024]	5.00 (1.06)
1.5	7.37 (1.55)	7.24 (1.48)	7.50	7.53	7.55	–	7.55	7.58 [512]	7.27 (1.53)
2.0	9.54 (1.97)	9.98	10.24	10.33	10.43	9.30	10.55	10.31 [1024]	9.08 (2.02)
2.5	12.51 (2.52)	12.63	12.99	13.07	13.23	–	13.54	13.02 [1024]	10.35 (2.54)
3.0	15.07	15.33	15.73	15.84	16.01	14.62	16.56	15.66 [512]	11.04 (3.02)

Table 3.a: SNR (dB) of SVQs Designed Using Algorithm 1 for a Generalized Gaussian Source with $\alpha = 2$ at Various Rates (bits/sample) and Block-Lengths. The Numbers in Parentheses Indicate the Actual Rate if Different from the Desired Rate. The Numbers in Brackets Indicate the LBG VQ Codebook Size.

$\alpha = 1.0$									
Rate	Block-Length					LMQ	ECSQ	LBG	PC
	4	8	16	24	32			VQ	
1.0	3.69 (0.79)	4.69 (0.90)	5.32 (0.97)	5.59	5.60	3.01	5.76	5.41 [1024]	5.25 (1.05)
1.5	6.81 (1.55)	7.32 (1.46)	8.05 (1.52)	8.15	8.25	–	8.55	7.72 [512]	6.83 (1.54)
2.0	9.29 (2.01)	9.72	10.39	10.54	10.72	7.53	11.31	10.30 [1024]	8.19 (2.05)
2.5	11.57 (2.52)	12.22	12.96	13.14	13.31	–	14.32	12.82 [1024]	8.78 (2.48)
3.0	14.15	14.92	15.68	15.88	16.03	12.61	17.20	15.20 [512]	9.17 (3.09)

Table 3.b: SNR (dB) of SVQs Designed Using Algorithm 1 for a Generalized Gaussian Source with $\alpha = 1$ at Various Rates (bits/sample) and Block-Lengths. The Numbers in Parentheses Indicate the Actual Rate if Different from the Desired Rate. The Numbers in Brackets Indicate the LBG VQ Codebook Size.

$\alpha = 0.5$									
Rate	Block-Length					LMQ	ECSQ	LBG	PC
	4	8	16	24	32			VQ	
1.0	5.49 (1.02)	7.19 (1.08)	7.28	7.51	7.90 (1.03)	1.82	8.53	7.16 [1024]	4.85 (1.04)
1.5	7.76 (1.46)	9.28	9.96	10.57	10.74	–	11.57	9.27 [512]	5.48 (1.48)
2.0	10.27 (2.07)	11.82 (2.02)	12.52	13.10	13.41	5.53	14.53	11.78 [1024]	5.85 (2.05)
2.5	11.92 (2.52)	14.02	14.96	15.36	15.79	–	17.49	14.06 [1024]	6.00
3.0	14.53 (3.01)	16.21	17.20	17.82	18.17	10.21	20.41	16.00 [512]	6.06 (3.01)

Table 3.c: SNR (dB) of SVQs Designed Using Algorithm 1 for a Generalized Gaussian Source with $\alpha = 0.5$ at Various Rates (bits/sample) and Block-Lengths. The Numbers in Parentheses Indicate the Actual Rate if Different from the Desired Rate. The Numbers in Brackets Indicate the LBG VQ Codebook Size.

$\alpha = 2.0$									
Rate	Block-Length					LMQ	ECSQ	LBG	PC
	4	8	16	24	32			VQ	
1.0	3.47 (0.79)	3.99 (0.87)	4.41 (0.94)	4.56 (0.97)	4.63	4.40	4.64	5.05 [1024]	5.00 (1.06)
1.5	7.01	7.23 (1.48)	7.46	7.55	7.57	–	7.55	7.58 [512]	7.27 (1.53)
2.0	9.57 (1.97)	10.11 (2.01)	10.33 (2.01)	10.35	10.40	9.30	10.55	10.31 [1024]	9.08 (2.02)
2.5	12.49 (2.51)	12.64 (2.48)	12.96	13.12	13.16	–	13.54	13.02 [1024]	10.35 (2.54)
3.0	15.33 (3.03)	15.53	15.83	15.84	15.97	14.62	16.56	15.66 [512]	11.04 (3.02)

Table 4.a: SNR (dB) of SVQs Designed Using Algorithm 2 for a Generalized Gaussian Source with $\alpha = 2$ at Various Rates (bits/sample) and Block-Lengths. The Numbers in Parentheses Indicate the Actual Rate if Different from the Desired Rate. The Numbers in Brackets Indicate the LBG VQ Codebook Size.

$\alpha = 1.0$									
Rate	Block-Length					LMQ	ECSQ	LBG VQ	PC
	4	8	16	24	32				
1.0	3.78 (0.79)	4.74 (0.88)	5.41 (0.98)	5.67	5.59	3.01	5.76	5.41 [1024]	5.25 (1.05)
1.5	6.94 (1.55)	7.58 (1.48)	7.87 (1.47)	8.17	8.26	–	8.55	7.72 [512]	6.83 (1.54)
2.0	9.38 (2.01)	10.03 (2.01)	10.54	10.75	10.87	7.53	11.31	10.30 [1024]	8.19 (2.05)
2.5	11.73 (2.51)	12.53	13.10	13.39	13.52	–	14.32	12.82 [1024]	8.78 (2.48)
3.0	13.99	15.18 (3.02)	15.65	15.99	16.14	12.61	17.20	15.20 [512]	9.17 (3.09)

Table 4.b: SNR (dB) of SVQs Designed Using Algorithm 2 for a Generalized Gaussian Source with $\alpha = 1$ at Various Rates (bits/sample) and Block-Lengths. The Numbers in Parentheses Indicate the Actual Rate if Different from the Desired Rate. The Numbers in Brackets Indicate the LBG VQ Codebook Size.

$\alpha = 0.5$									
Rate	Block-Length					LMQ	ECSQ	LBG VQ	PC
	4	8	16	24	32				
1.0	5.60 (1.02)	6.91 (1.07)	7.43	7.80	7.94 (1.03)	1.82	8.53	7.16 [1024]	4.85 (1.04)
1.5	7.38 (1.40)	9.27	10.19	10.62 (1.49)	10.95	–	11.57	9.27 [512]	5.48 (1.48)
2.0	10.35 (1.95)	12.02 (2.03)	12.78	13.28	13.58	5.53	14.53	11.78 [1024]	5.85 (2.05)
2.5	12.57 (2.48)	14.15	15.24	15.87	16.27	–	17.49	14.06 [1024]	6.00
3.0	15.20 (3.03)	16.56	17.67	18.23	18.53	10.21	20.41	16.00 [512]	6.06 (3.01)

Table 4.c: SNR (dB) of SVQs Designed Using Algorithm 2 for a Generalized Gaussian Source with $\alpha = 0.5$ at Various Rates (bits/sample) and Block-Lengths. The Numbers in Parentheses Indicate the Actual Rate if Different from the Desired Rate. The Numbers in Brackets Indicate the LBG VQ Codebook Size.

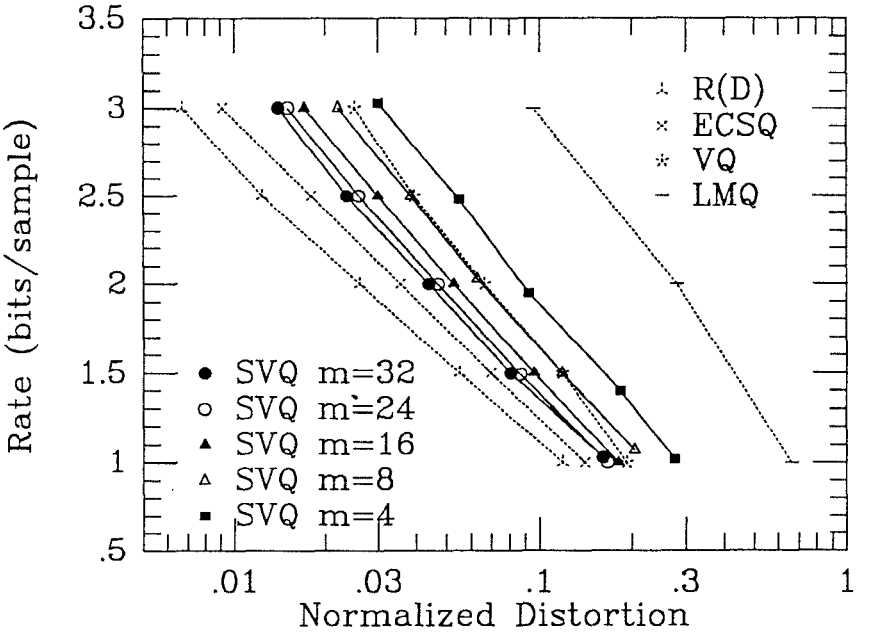

Fig. 1: Rate-Distortion Characteristics of the SVQ (Algorithm 2) for a Generalized Gaussian Source with $\alpha = 0.5$.

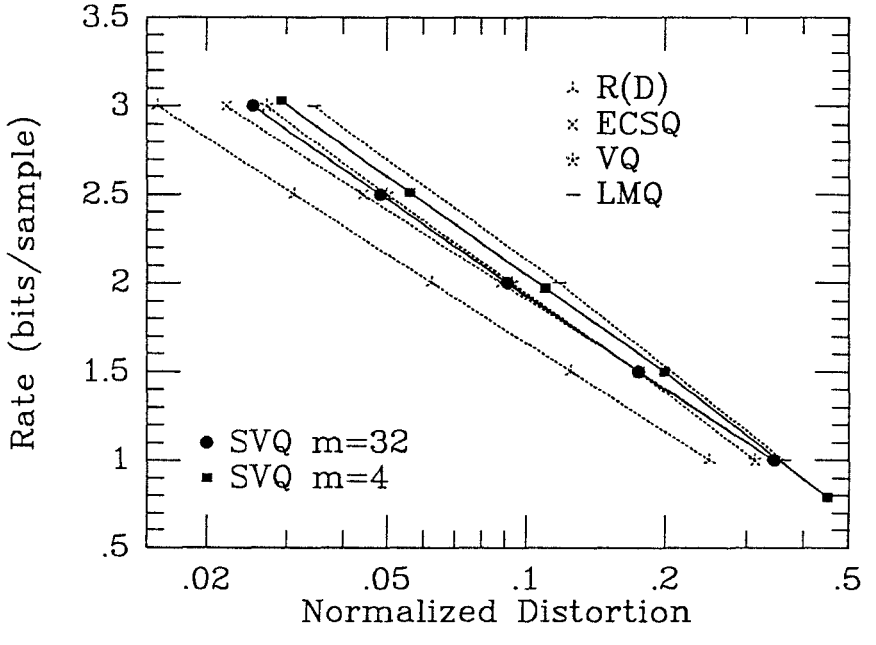

Fig. 2: Rate-Distortion Characteristics of the SVQ (Algorithm 2) for a Generalized Gaussian Source with $\alpha = 2$.

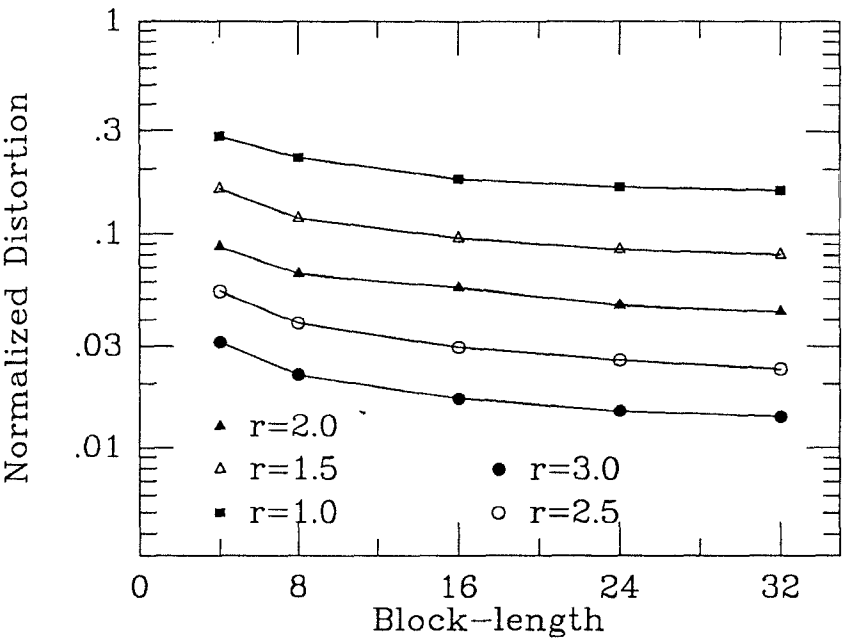

Fig. 3: Normalized Average Distortion of the SVQ (Algorithm 2) as a Function of Block-Length for a Generalized Gaussian Source with $\alpha = 0.5$.

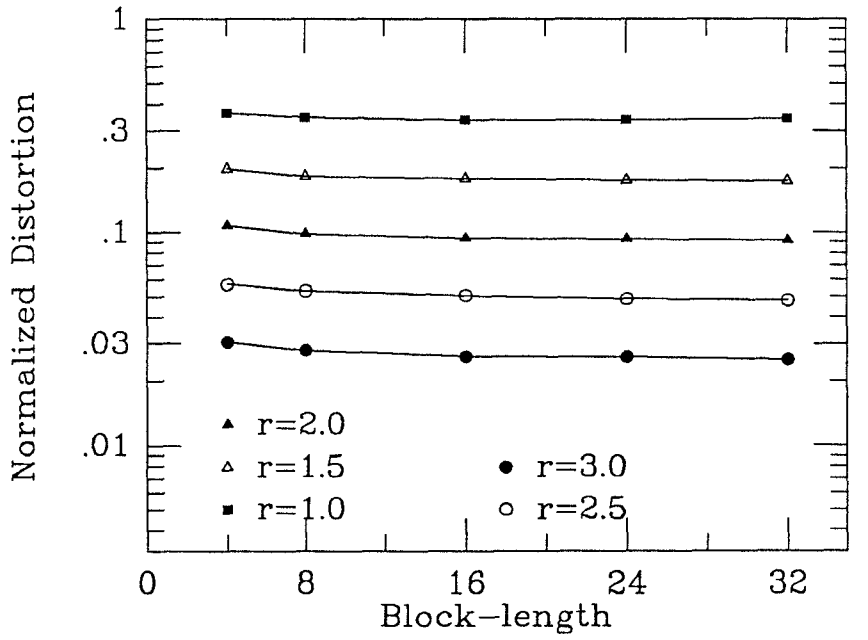

Fig. 4: Normalized Average Distortion of the SVQ (Algorithm 2) as a Function of Block-Length for a Generalized Gaussian Source with $\alpha = 2$.

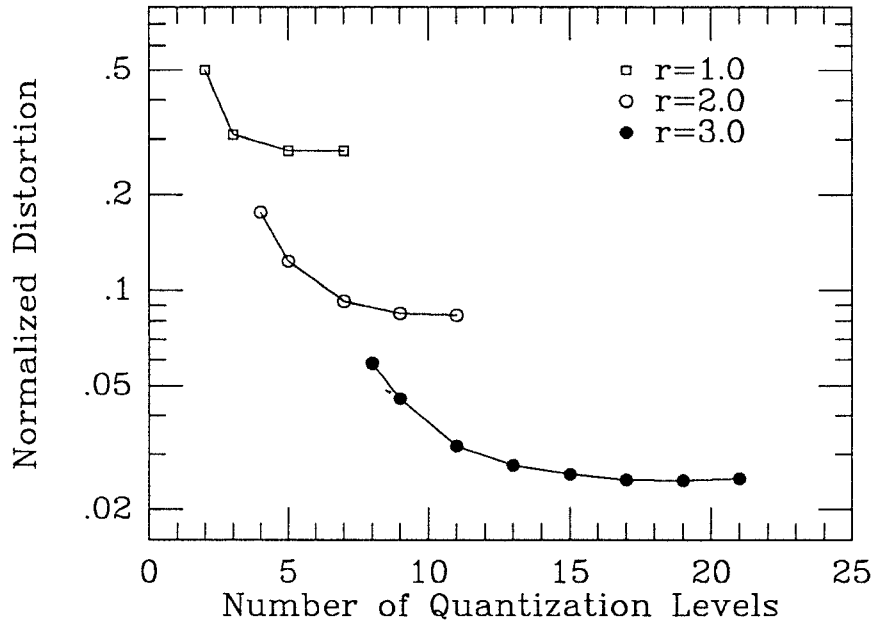

Fig. 5: Normalized Average Distortion of the SVQ (Algorithm 1) as a Function of the Number of Quantization Levels n for a Generalized Gaussian Source with $\alpha = 1$ and $m = 32$.

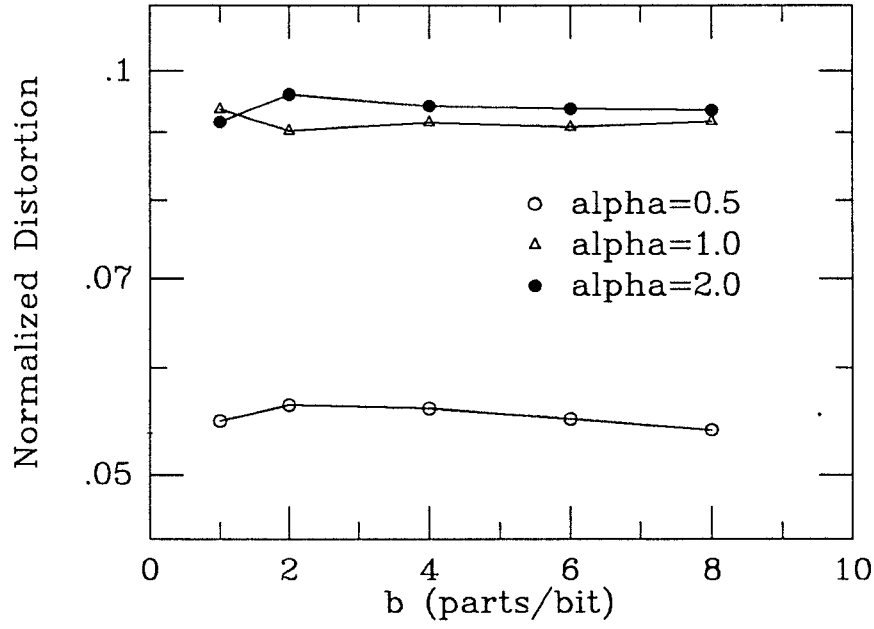

Fig. 6: Normalized Average Distortion of the SVQ (Algorithm 1) as a Function of b for $m = 16$ and $r = 2$ bits/sample.

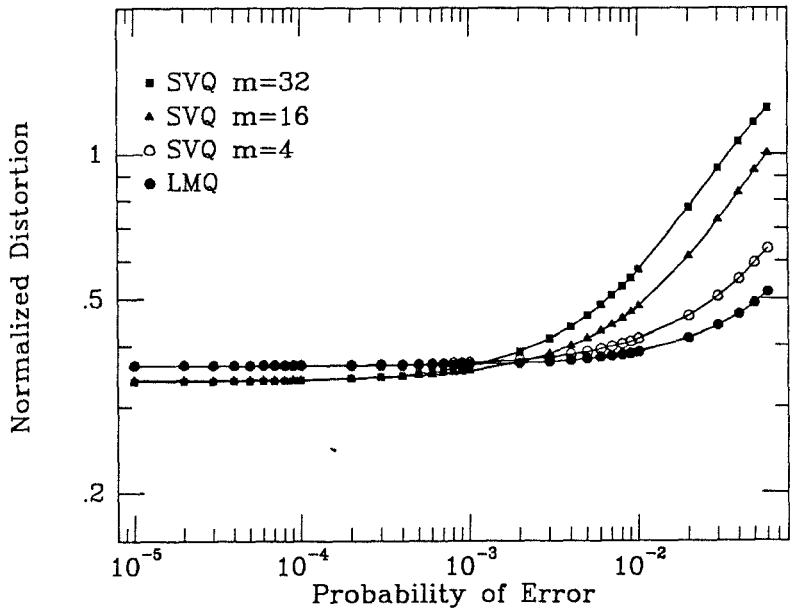

Fig. 7: Normalized Average Distortion of the SVQ and LMQ as a Function of Channel Bit-Error Probability for a Generalized Gaussian Source with $\alpha = 2$ and $r = 1$ bit/sample.

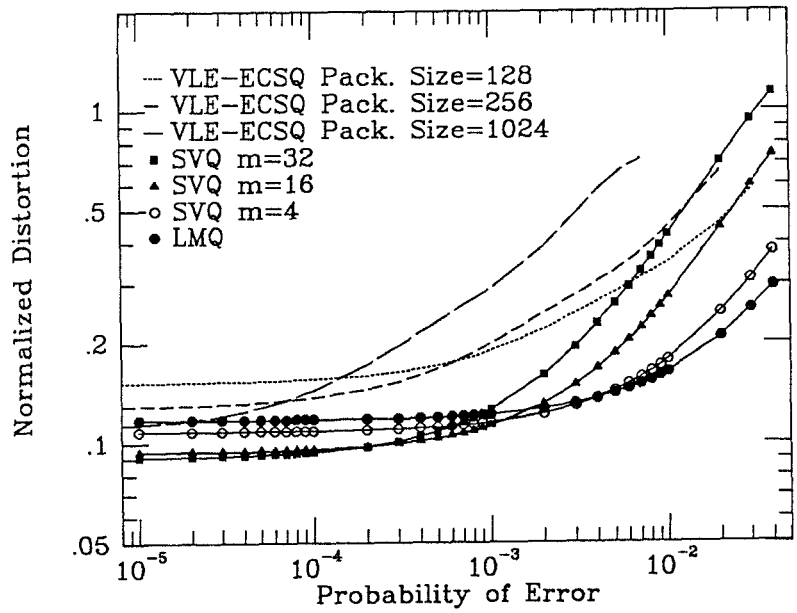

Fig. 8: Normalized Average Distortion of the SVQ, LMQ and Packetized VLE-ECSQ as a Function of Channel Bit-Error Probability for a Generalized Gaussian Source with $\alpha = 2$ and $r = 2$ bits/sample.

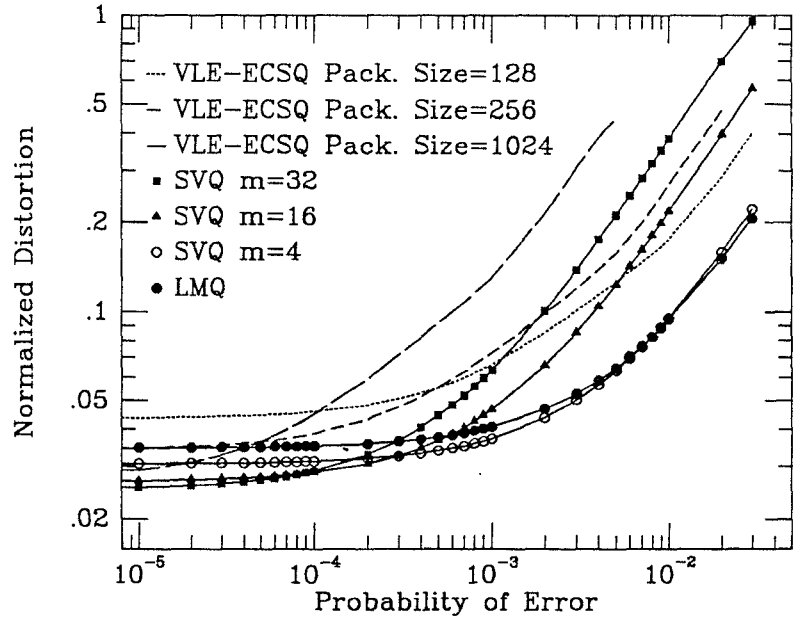

Fig. 9: Normalized Average Distortion of the SVQ, LMQ and Packetized VLE-ECSQ as a Function of Channel Bit-Error Probability for a Generalized Gaussian Source with $\alpha = 2$ and $r = 3$ bits/sample.

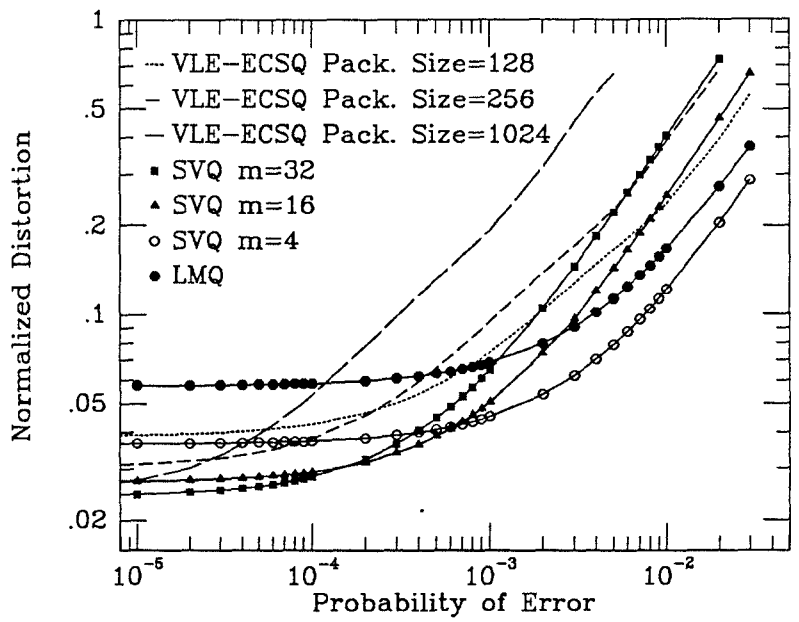

Fig. 10: Normalized Average Distortion of the SVQ, LMQ and Packetized VLE-ECSQ as a Function of Channel Bit-Error Probability for a Generalized Gaussian Source with $\alpha = 1$ and $r = 3$ bits/sample.

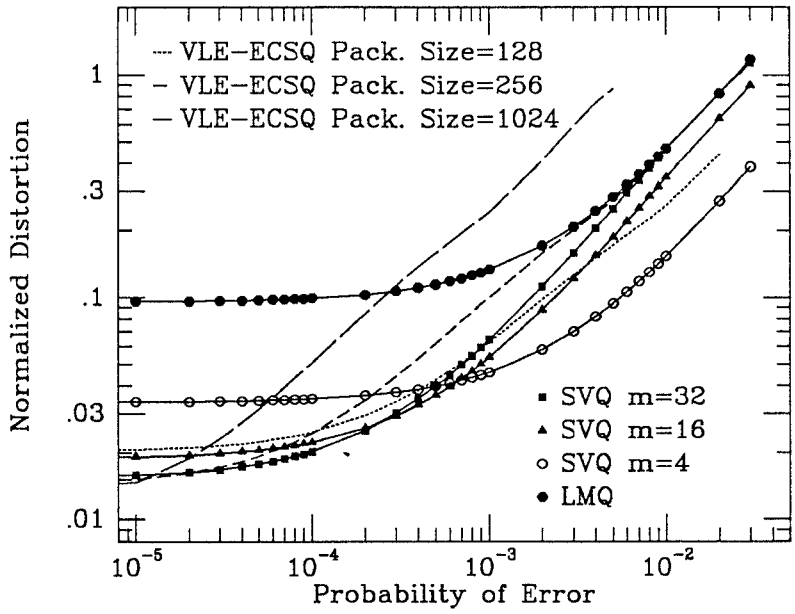

Fig. 11: Normalized Average Distortion of the SVQ, LMQ and Packetized VLE-ECSQ as a Function of Channel Bit-Error Probability for a Generalized Gaussian Source with $\alpha = 0.5$ and $r = 3$ bits/sample.

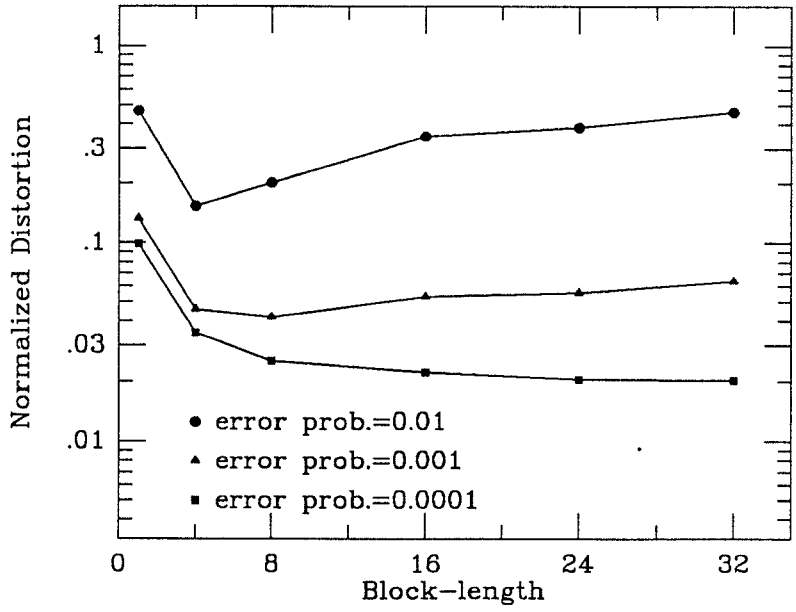

Fig. 12: Normalized Distortion of the SVQ as a Function of the Block-Length (Noisy Channel) for a Generalized Gaussian Source with $\alpha = 0.5$ and $r = 3$ bits/sample.

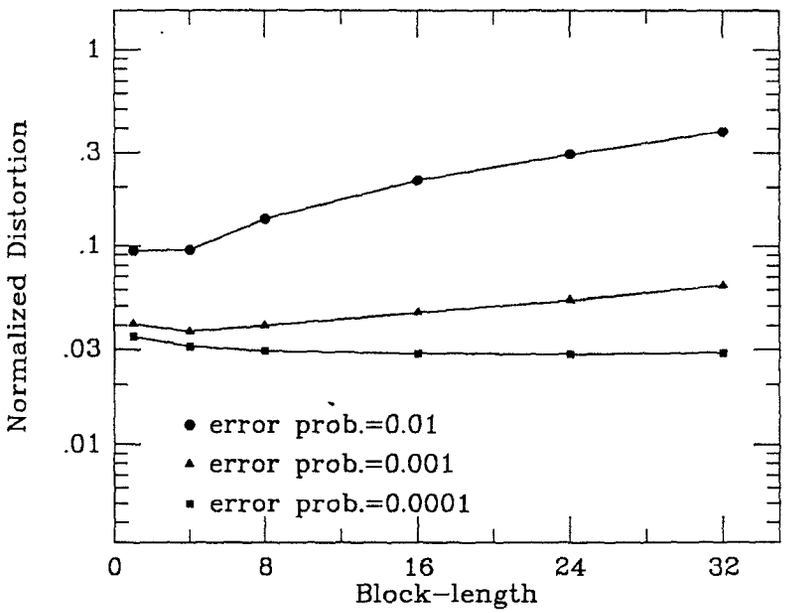

Fig. 13: Normalized Distortion of the SVQ as a Function of the Block-Length (Noisy Channel) for a Generalized Gaussian Source with $\alpha = 2$ and $r = 3$ bits/sample.