A Structured Fixed-Rate Vector Quantizer Derived from a Variable-Length Scalar Quantizer: Part II—Vector Sources

Rajiv Laroia, Member, IEEE, and Nariman Farvardin, Member, IEEE

Abstract—The fixed-rate scalar-vector quantizer (SVQ) was described in Part I for quantizing stationary memoryless sources. In this sequel, the SVQ has been extended to a specific type of vector sources in which each component is a stationary memoryless scalar subsource independent of the other components. Algorithms for the design and implementation of the original SVQ are modified to apply to this case. The resulting SVQ, referred to as the extended SVQ (ESVQ), is then used to quantize stationary sources with memory (with known autocorrelation function). This is done by first using a linear orthonormal block transformation, such as the Karhunen-Loève transform, to decorrelate a block of source samples. The transform output vectors, which can be approximated as the output of an independent-component vector source, are then quantized using the ESVQ. Numerical results are presented for the quantization of first-order Gauss-Markov sources using this scheme. It is shown that the ESVQ-based scheme performs very close to the entropy-coded transform quantization while maintaining a fixed-rate output and outperforms the fixed-rate scheme that uses scalar Lloyd–Max quantization of the transform coefficients. Finally, it is shown that this scheme also performs better than implementable vector quantizers, specially at high rates.

Index Terms—Fixed-rate coding, structured vector quantization, scalar-vector quantizer, independent-component vector sources, correlated Gaussian sources.

I. INTRODUCTION

The fixed-rate scalar-vector quantizer (SVQ) for quantizing stationary memoryless scalar sources was introduced in Part I of this paper [1]. The SVQ is derived from a variable-length scalar quantizer and bridges the gap between the performance of the optimal entropy-constrained scalar quantizer [2]–[4] and the error-minimizing fixed-rate Lloyd–Max quantizer (LMQ) [5], [6] while maintaining a fixed-rate output. The design and implementation complexities of the SVQ are only polynomial in block length and are manageable even for relatively fine quantization (high bit rates) at large block lengths. For memoryless sources, the SVQ is an attractive alternative to LBG vector quantizers (LBG VQ) [7]. This is especially true at high rates where, for comparable performance, the LBG VQ complexity (which grows exponentially with the rate-dimension product) is significantly higher than the SVQ complexity.

In this paper, the SVQ has been extended to the quantization of a specific type of vector sources and the resulting quantizer is referred to as the extended SVQ (ESVQ). Each component of the vector source considered here is a stationary memoryless scalar subsource independent of the other components; the scalar subsources can have distinct probability distributions. Such sources are encountered in the transform coding of images, speech and other signals. An algorithm for the suboptimal design of the ESVQ is developed along with fast algorithms for codebook search and codevector encoding/decoding.

The ESVQ for independent-component vector sources is then used to quantized stationary sources with memory. This is done in the context of block transform quantization. The idea of block transform quantization was introduced by Huang and Schultheiss (HS) in [8] for quantizing stationary correlated Gaussian sources. They use the Karhunen–Loève transform, which is a decorrelating orthonormal linear transformation, to decorrelate a block of source samples. Each transform output vector is quantized using a bank of fixed-rate Lloyd–Max scalar quantizers. Farvardin and Lin in [9] have improved upon these results (in the rate-distortion sense) by using a bank a variable-length encoded scalar quantizers in place of LMQ's. The price of this improvement is a variable-rate output that makes the system more complex to implement because of the buffer overflow/underflow and error propagation problems associated with variable-rate systems. Since the output vectors of the decorrelating transform can be approximated as a block of source samples, each transform output vector is quantized using a bank of fixed-rate Lloyd–Max scalar quantizers. Farvardin and Lin in [9] have improved upon these results (in the rate-distortion sense) by using a bank a variable-length encoded scalar quantizers in place of LMQ's. The price of this improvement is a variable-rate output that makes the system more complex to implement because of the buffer overflow/underflow and error propagation problems associated with variable-rate systems. Since the output vectors of the decorrelating transform can be approximated as a block of source samples, each transform output vector is quantized using a bank of fixed-rate Lloyd–Max scalar quantizers. Farvardin and Lin in [9] have improved upon these results (in the rate-distortion sense) by using a bank a variable-length encoded scalar quantizers in place of LMQ's. The price of this improvement is a variable-rate output that makes the system more complex to implement because of the buffer overflow/underflow and error propagation problems associated with variable-rate systems.

Since an ESVQ-based system enjoys the same structural properties as the SVQ, its computational complexity is significantly lower than that of an LBG VQ for the same codebook size. Therefore, with the ESVQ-based system, dramatically larger codebooks can be used before the complexity or memory requirements render it impractical. This means that for a given encoding rate, the block length of the ESVQ-based system can be much larger than that of the LBG VQ, and hence, smaller distortions may be achieved (especially if the source...
is heavily correlated). Specific examples given for stationary Gauss–Markov sources will confirm this claim.

This paper is organized as follows. The extension of the SVQ to independent-component vector sources (ESVQ) is considered in Section II. An algorithm for a suboptimal design of the ESVQ is presented in Section III while Section IV addresses the implementation issues which include codebook search and codevector encoding. Quantization of stationary sources with memory is considered in Section V along with numerical results for quantizing first-order Gauss–Markov sources and appropriate comparisons with several other schemes. In Section VI, the ESVQ is compared with an SVQ-based system for quantizing independent-component vector sources. Finally, the summary and conclusions are presented in Section VII.

II. EXTENSION TO INDEPENDENT-COMPONENT VECTOR SOURCES (ESVQ)

In this section, we extend the SVQ for memoryless sources to a specific type of vector sources. The vector sources considered here are assumed to have independent components, and each component is assumed to be a stationary memoryless scalar subsource; each scalar subsource can have a distinct probability distribution. In the special case when all subsources are identical, the ESVQ reduces to the SVQ for memoryless scalar sources. Extension of the SVQ to independent-component vector sources paves the way for the development of a scheme for the quantization of vector sources with dependent components, as we will show later.

Since the various components of the previously mentioned vector source can have different distributions, an ESVQ for an m-dimensional vector source is derived from m different variable-length scalar quantizers $S_1, S_2, \ldots, S_m$—each tailored to the distribution of the corresponding subsource. Let $Q_i \equiv \{q_{i1}, q_{i2}, \ldots, q_{in_i}\}$ denote the set of quantization levels of the $i$th quantizer in the $S_i$ for $i \in J_m = \{1, 2, \ldots, m\}$ and $L_i \equiv \{l_{i1}, l_{i2}, \ldots, l_{in_i}\}$ be the corresponding set of positive integer lengths assigned to the quantization levels. Also, let $Q \equiv Q_1 \cup Q_2 \cup \cdots \cup Q_m$ and $L \equiv L_1 \cup L_2 \cup \cdots \cup L_m$ denote the set of all quantization levels and all lengths, respectively. The ESVQ is defined as follows.

**Definition 1:** An m-dimensional ESVQ $V$ derived from scalar quantizers $S_1, S_2, \ldots, S_m$ is a vector quantizer with a codebook $Z$ given as

$$Z = \{z = (z_1, z_2, \ldots, z_m) \in Q_1 \times Q_2 \times \cdots \times Q_m :$$

$$\ell_f(z_{j1}) + \ell_f(z_{j2}) + \cdots + \ell_f(z_{jm}) \leq L\},$$

where the threshold $L$ governs the rate $mr$ (in bits per m-dimensional vector) of the quantizer, and the index function $f_j : Q_j \rightarrow J_m$ is defined by $f_j(q_{ij}) = j$, $j \in J_{n_i}, i \in J_m$.

**Remark 1:** According to Definition 1, the ESVQ is unchanged if: a) an integer $k_i$ is added to all the lengths in $L_i$, $i \in J_m$ (assuming that the resulting lengths are still positive) and $k = \sum_{i \in J_m} k_i$ is added to the threshold $L$; and b) all the lengths in $L = L_1 \cup L_2 \cup \cdots \cup L_m$ are multiplied by a positive rational $s$ (assuming that the resulting lengths are still integers) and the threshold $L$ is also multiplied by $s$.

**Remark 2:** It is interesting to note that in the case of ESVQ, a permutation of any codevector in general does not result in a codevector. This is because the different components of the codevector belong to different alphabets. Hence, the idea of scalar-vector quantization can be extended to sources for which permutation coding does not work well.

**Remark 3:** Weighted pyramid and elliptical vector quantizers [11] are extensions of the pyramid and spherical vector quantizers [10], respectively, to independent-component vector sources. The scalar subsources of the vector source in this case have different variances but same distributions (Laplacian for weighted pyramid and Gaussian for elliptical vector quantizers). The weighted pyramid vector quantizer is essentially a special case of the ESVQ when all the scalar quantizers are uniform and $\ell_f(z) = w_i z_i$, $j \in J_{n_i}, i \in J_m$. Here, $w_i$ is the weight associated with the $i$th subsource. The elliptical vector quantizer is similarly a special case of the ESVQ when all scalar quantizers are uniform and $\ell_f(z) = w_i z_i$.

A procedure for the design of the ESVQ is presented in the next section. Some of the steps in this case are straightforward extensions of the SVQ [1], [12], but are presented here for the sake of completeness.

III. DESIGN OF THE ESVQ

Designing a rate $r$ bits/sample ESVQ for a given $m$-dimensional independent-component vector source amounts to determining the quantization levels $Q$, the lengths $L$ and the threshold $L$. As in the case of the SVQ [1], this can be done by a two-step iterative process. The first step (Step A) determines the quantization levels $Q$ given $L$ and $L$. The second step (Step B) determines the lengths $L$ and threshold $L$ given $Q$. If both these steps are optimal with respect to some distortion measure, then each iteration reduces the average distortion, and convergence to a locally optimal design is guaranteed. While Step A can be optimally performed for some important distortion metrics, the optimal solution of Step B is not known and a suboptimal solution is presented here. This does not guarantee the convergence of the design algorithm but results in good suboptimal designs. The performance of the ESVQ designed using this algorithm is sensitive to the initial choice of $Q, L,$ and $L$ in the iterative process. For the quantizers designed here, we start with uniformly spaced quantization levels for each $Q_i, i \in J_m$. The spacing between the levels was chosen such that the sum of the output entropies of $S_1, S_2, \ldots, S_m$ (assuming quantization thresholds to be midway between the levels) equals the rate $mr$ bits/vector. For this choice of $Q$ the initial values of $L$ and $L$ were assigned in accordance with Step B.

**A. Step A**

In this step, we determine $Q$ for a given $L$ and $L$. The procedure used here is a simple extension of that for the SVQ [1] and is itself performed iteratively. The nonnegative, single-letter distortion measure between the source output $x$ and its reproduction $y$ is denoted by $d(x, y)$. A long training set of $N$ source vectors is generated and this set is used to characterize the $m$-dimensional probability density function (pdf) of the
vector source. Let \( \mathcal{X} = \{ x_j, j \in J_N \} \) denote this set where \( x_j = (x_{ij}, x_{2j}, \ldots, x_{mj}) \).

**Algorithm**

0) For a given \( (C, L) \) the training sequence \( \mathcal{X} \) and stopping threshold \( \epsilon > 0 \), select as the initial set of quantization levels \( Q^0 = \{ Q_1^0, Q_2^0, \ldots, Q_m^0 \} \), the optimal solution for \( Q \) in the previous visit to Step A, and set the iteration index \( k = 0 \). The procedure for choosing \( Q^0 \) for the first visit to Step A has already been described.

1) Quantize each vector in \( \mathcal{X} \) using the ESQ definition by \( (Q^k, L, C) \). Let \( \mathcal{Y}_k = \{ y_{ij}^k, j \in J_N \} \) represent the resulting set of quantized vectors where \( y_{ij}^k = (y_{ij1}^k, y_{ij2}^k, \ldots, y_{ijm}^k) \in Q_1^k \times Q_2^k \times \cdots \times Q_m^k \), for \( j \in J_N \). The average per-vector distortion \( D^k \) associated with this quantization operation can be expressed as

\[
D^k = \frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{m} d(x_{ij}, y_{ij}^k)
\]

Thus, for the squared-error distortion measure, \( D^k \) is a generalization of the asymptotically optimal (for large \( m \)) solution of \( L \) for the SVQ (Method 2 in [1]).

2) Update the set of quantization levels to \( Q^{k+1} \) where

\[
q_{hi}^{k+1} = \arg \min_q \sum_{j: y_{ij}^k = q_{hi}} d(x_{ij}, q), \quad l \in J_m, \quad i \in J_N
\]

It is straightforward to show that \( Q^{k+1} \) minimizes \( D^k \) and hence reduces distortion over \( Q^k \). In the important special case of squared-error distortion \( d(x, y) = (x - y)^2 \), it is easily shown that (3a) reduces to

\[
q_{hi}^{k+1} = \left\{ \sum_{y_{ij}^k = q_{hi}} 1 \right\}^{-1} \left[ \sum_{y_{ij}^k = q_{hi}} x_{ij} \right] \quad l \in J_m, \quad i \in J_N
\]

Thus, for the squared-error distortion measure, (3b) corresponds to taking all the input samples that quantize to \( q_{hi}^k \) (using the ESVQ defined by \( (Q^k, L, C) \)) and updating this quantization level to \( q_{hi}^{k+1} \), which is the mean of all those samples. Similarly, for the absolute-error distortion \( d(x, y) = |x - y| \), each quantization level \( q_{hi}^k \) is updated to the median of all the input samples that are quantized to it.

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

Since the updating in Step 2) can only reduce the average distortion, the above algorithm converges for a given \( (L, C) \); resulting in \( Q^{opt}(L, C) \).

**B. Step B**

In this step, we determine \( L \) and \( C \) for a given \( Q \). As mentioned before, the optimal solution for this step is not known. In what follows, we present a suboptimal solution for determining \( L \). After obtaining \( L \), the threshold \( C \) can be chosen such that the ESVQ has the desired output rate.

1) Determination of \( C \): For the SVQ design, two methods for determining \( C \) given \( Q \) are described in [1]. While the first method (Method 1) performs well for the SVQ, there is an important reason for not using a similar method for the ESVQ. The lengths in [1], for a given \( C \), are assigned based on probabilities of quantization to the nearest level using the scalar quantizer \( S \). This corresponds to the quantization thresholds being placed midway between the quantization levels of \( S \). A similar procedure for ESVQ will result in a set of lengths \( L_i \) of the quantizer \( S_i \) that is determined only by the probability distribution of the \( i \)th component. Hence, the lengths associated with one quantizer are not dependent upon those of any other quantizer. This implies that there is no way of placing greater importance on the quantization of some (possibly larger variance) components of the vector source over the others. As an example, consider a two-dimensional source having zero-mean Gaussian components with variances 1 and 100. Also let \( Q_1 = \{ q_{11}, q_{12}, \ldots, q_{1n} \} \) and \( Q_2 = \{ 10q_{11}, 10q_{12}, \ldots, 10q_{1n} \} \). This will result in identical sets of lengths \( L_1 \) and \( L_2 \). Clearly, a more desirable length assignment is one in which most length values in \( L_1 \) are relatively large. To overcome this problem we use the following algorithm that is a generalization of the asymptotically optimal (for large \( m \)) solution of \( L \) for the SVQ (Method 2 in [1]).

First, we determine the set of quantization thresholds \( T_i \equiv \{ t_{i0}, t_{i1}, \ldots, t_{im} \} \) for each scalar quantizer \( S_i \) with quantization levels \( Q_i \), such that the expected distortion is minimized given that the sum of the output entropies of the quantizers is no greater than the rate \( mr \) bits/vector. As described later, the probabilities of the quantization regions defined by these thresholds are used to determine the lengths in \( L \). Hence, the problem here is to determine the set of quantization thresholds \( T \equiv \{ T_1 \cup T_2 \cup \cdots \cup T_m \} \) such that the distortion

\[
D(T) = \sum_{i=1}^{m} \sum_{j=1}^{n_i} \int t_{ij}^{m_{ij}} d(x, q_{ij}) p_i(x) dx
\]

is minimized subject to the constraint on the output entropy

\[
H(T) = -\sum_{i=1}^{m} \sum_{j=1}^{n_i} p_{ij} \log_2 p_{ij} \leq mr
\]
the quantization thresholds are determined the lengths \( \ell_j \) are evaluated as
\[
\ell_j = \lfloor b \log_2 (1/p_j) \rfloor, \quad j \in J_{m1}, \quad i \in J_m, \tag{5}
\]
where the square brackets denote rounding off to the nearest integer and \( b \) is a real number that determines the effective round-off error. Larger \( b \) implies a smaller effective error but a higher implementation complexity. The effect of \( b \) on the complexity and performance of the SVQ is discussed in [1] and similar arguments hold for the ESVO.

Even though the ESVO does not explicitly allocate bits among the various components of the vector source, the bit (entropy) allocation in the above algorithm is implicit in the way the lengths are assigned to the quantization levels of the different scalar quantizers.

2) Determination of the Threshold \( L \): Now that \( L \) has been determined, the threshold \( L \) is obtained by counting the grid-points (starting with the ones that have the smallest total length) until \( 2^{m_r} \) points are counted; the largest total length for which all grid-points of that length are included in this collection will determine the value of \( L \). An algorithm to accomplish this is described below.

Let \( N_{i}^{f} \) represent the number of distinct \( i \)-vectors \((v_{1}, v_{2}, \ldots, v_{i}) \in Q_{1} \times Q_{2} \times \cdots \times Q_{i} \) such that their total length \( \ell_{f}(v_{1}) + \ell_{f}(v_{2}) + \cdots + \ell_{f}(v_{i}) = j \). Then, \( N_{i}^{f} \) satisfies the recursive equation
\[
N_{i}^{f} = \sum_{i=1}^{j} N_{i-1}^{f-1}, \quad i \in J_{m} \quad \text{where} \quad N_{i}^{f} = 0 \quad \text{for} \quad j < 0 \quad \text{and} \quad N_{0}^{f} = 1.
\]
The number \( C_{m}^{f} \) of grid-points with a total length less than or equal to \( j \) is now given by \( C_{m}^{f} = \sum_{i=1}^{j} N_{i}^{f} \). The threshold \( L \) can therefore be evaluated as,
\[
L = \max \{ j : C_{m}^{f} \leq 2^{m_r} \}.
\]
This choice \( L \) guarantees that there will be at most \( 2^{m_r} \) vectors in the codebook and hence each codevector can be encoded by an \( m_r \)-bit binary codeword.

C. ESVO Design Algorithm

The ESVO design consists of successive updating of \( Q \) and \((L, L)\) by applying Steps A and B, respectively. As noted before, the convergence of this algorithm is not guaranteed because of the suboptimality of Step B. For cases where the solution does not converge, a fixed number of iterations are performed from which the best quantizer is chosen. As with the SVQ, for those cases where the solution converged, the convergence does not imply any optimality or local optimality and can be attributed to the discrete (integer) nature of the lengths in \( L \). The design algorithm for the ESVO is presented next.

Algorithm

0) Let \( \epsilon > 0 \), \( k_{\text{max}} \) and \( k_{\text{min}} \) denote the stopping threshold, the maximum and the minimum number of iterations. For a given training sequence \( X \), design an initial ESVO consisting of \( Q^{0}, L^{0} \), and \( L^{0} \) as explained in the beginning of this section; set the iteration index \( k = 0 \); set initial average distortion \( D^{0} = \infty \).

1) For fixed \( L^{k} \) and \( L^{k} \), use Step A (with \( Q^{k} \) as initial set of quantization levels) to obtain \( Q^{k+1} = Q^{k+1}(L^{k}, L^{k}) \). Compute the average distortion \( D^{k+1} \) associated with the triple \((Q^{k+1}, L^{k}, L^{k})\). If \((D^{k} - D^{k+1})/D^{k} < \epsilon \) and \( k \geq k_{\text{min}} \) stop with \((Q^{k+1}, L^{k}, L^{k}) \) defining the ESVO; else if \( k = k_{\text{max}} \), stop and select the ESVO as that which resulted in the smallest average distortion among all iterations up to \( k_{\text{max}} \); else, set \( k = k + 1 \).

2) For \( Q^{k} \), determine from Step B, the set of lengths \( L^{k} \) and the threshold \( L^{k} \). Go to Step 1).

IV. IMPLEMENTATION OF THE ESVO

The implementation of the ESVO defined by \( Q \), \( L \) and \( L \) essentially involves two steps: 1) finding the vector in the codebook closest to the input vector (codebook search) and 2) assignment of a binary codeword to the selected codevector at the transmitter and finding the codevector corresponding to the received codeword at the receiver (codevector encoding/decoding). Given the special structure of the ESVO codebook, both these steps can be efficiently performed as described next.

A. Codebook Search

Let \( D^{i}_{t} \) be the minimum squared-error distortion that results when the first \( i \) components of the input \( m \)-vector \( x = (x_{1}, x_{2}, \ldots, x_{m}) \) are quantized to any of the \( i \)-vectors \( z_{1} = (z_{1}, z_{2}, \ldots, z_{i}) \in Q_{1} \times Q_{2} \times \cdots \times Q_{i} \) such that the total length of \( z_{i} \) is \( j \), that is, \( \ell_{f}(z_{1}) + \ell_{f}(z_{2}) + \cdots + \ell_{f}(z_{i}) = j \).

Also, let \( z_{j}^{i} \) represent the \( i \)-vector that results in the minimum distortion \( D^{i}_{t} \). The distortion \( D^{i+1}_{t} \) is then recursively given as
\[
D^{i+1}_{t} = \min_{k \in J_{a(i+1)}} \left[ D^{i}_{t} - \delta(x_{i+1} - q_{t}(x_{i+1})) \right]^{2}, \quad i \in \{0\} \cup J_{m-1}, \tag{6}
\]
where \( D^{i}_{t} = \infty \) for \( j \leq 0 \) or \( i \leq 0 \) but \( D^{0}_{t} = 0 \).

If the above is minimized for \( k = k' \), then the corresponding minimum distortion vector \( x_{j+1}^{i+1} \) is given by the recursive equation \( x_{j+1}^{i+1} = \left( x_{j+1}^{i+1} - q_{t}(x_{j+1}) \right) \). We can solve (6) using a dynamic programming algorithm [13] similar to the Viterbi algorithm [14] and determine \( D^{m}_{t} \) for all \( j \in J_{L} \). The minimum distortion \( D_{\text{min}} \) is then given as, \( D_{\text{min}} = \min_{j \in J_{L}} D^{m}_{t} \).

If \( j = j' \) minimizes this distortion, then \( x_{j}^{m} \) gives the desired codevector.

Although the order in which indexes are assigned to the scalar subsources to form a vector source does not affect the performance of the ESVO, it does make a difference on its implementation complexity. From an implementation standpoint, it is advantageous to label the subsources such that their variances are in a decreasing order, i.e., the first component of the vector has the largest variance, the second component has the next largest variance and so on. This is because a large variance subsource (ith) typically has a large number \( n_{i} \) of quantization levels associated with it. If such a subsource is assigned a smaller index, (6) for this source will have to be computed for a smaller range of values of \( j \), resulting in reduced implementation complexity.

The codebook search algorithm previously described has a considerable higher complexity than simple component-by-
component scalar quantization of the source \( m \)-vector using the scalar quantizer bank \( S_1, S_2, \ldots, S_m \). The overall search complexity can hence be reduced by first performing this simple component-by-component quantization and checking the total length of the resulting vector (grid-point). If the total length is no greater than the threshold \( L \), this is the required codevector; if not, the codebook search algorithm can be used.

**B. Encoding of Codevectors**

There are \( 2^m \) codevectors in the codebook \( Z \) of the ESVQ and the encoder is a mapping that assigns \( m \)-bit binary numbers to the codevectors in a one-to-one manner. The following algorithm implements one such mapping.

To every codevector \( \mathbf{x} = (x_1, x_2, \ldots, x_m) \in Z \) assigned a unique \( m \)-tuple \( N(\mathbf{z}) \equiv (f_m(z_m), f_{m-1}(z_{m-1}), \ldots, f_1(z_1)) \). All the possible \( m \)-tuples can now be ordered in a dictionary order. That is, one \( m \)-tuple is smaller than the other if either its first component is smaller than the first component of the other, or if their first components are equal and the second component is smaller than the second component of the other, and so on. The encoder function \( E : Z \rightarrow \{0, 1, \ldots, 2^m - 1\} \) is defined as \( E(\mathbf{x}) = \sum_{l=1}^{m} 2^{l-1} f_l(\mathbf{x}_l) \), where \( \mathbf{x}_l \) is the \( l \)-th component of \( \mathbf{x} \), and \( E(\mathbf{z}) = 0 \) if \( N(\mathbf{z}) < N(\mathbf{x}) \), and \( E(\mathbf{z}) = m \) if \( N(\mathbf{z}) > N(\mathbf{x}) \). In other words, \( E(\mathbf{x}) \) is the number of codevectors in \( Z \) that are "smaller" than \( \mathbf{x} \) (smaller in the sense \( N(\mathbf{w}) < N(\mathbf{z}) \)).

We can also write \( E(\mathbf{x}) = \sum_{k=1}^{m} E_k(\mathbf{x}_k) \), where \( E_k(\mathbf{x}_k) \) is the number of codevectors \( \mathbf{w} \) such that the \( m \)-tuples \( N(\mathbf{w}) \) and \( N(\mathbf{z}) \) are equal in their first \((k-1)\) components while the \( k \)-th component of the \( \mathbf{w} \)-tuple is smaller than that of the \( \mathbf{z} \)-tuple. The \( m \)-bit binary codeword associated with the codevector \( \mathbf{z} \) is then the binary representation of \( E(\mathbf{z}) \).

Let \( C_l^m \) represent the number of distinct \( l \)-vectors \((z_1, z_2, \ldots, z_l) \in Q_1 \times Q_2 \times \cdots \times Q_l \) with total length no greater than \( j \). Also, define \( C_l^j = 0 \) if \( j < 0 \) and \( 1 \), otherwise. It is now easy to see that \( C_l^j = \sum_{j=0}^{\infty} \binom{m-j}{l} (2^{m-j} - 2^j) \), for \( k \in J_m \), where \( \mathbf{L}(\mathbf{x}) = \sum_{l=1}^{m} L_l \) and \( \mathbf{L}_i = \sum_{j=m-i+1}^{m} L_j(\mathbf{x}_j) \), \( i \in J_m \).

Now, we can write \( E(\mathbf{x}) \) as

\[
E(\mathbf{x}) = \sum_{k=1}^{m} \sum_{l=1}^{j} C_{m-k}^{l-1} f_k(\mathbf{x}_k).
\]

Note that \( C_l^m \) can be evaluated for \( i \in J_m \) and \( j > 0 \) by expressing it in terms of \( N_l^{m} \) (Section III) as \( C_l^j = \sum_{l=1}^{m} N_l^{m} \).

For a fast implementation of this algorithm, the \( C_l^m \) can be evaluated once and stored in memory.

The decoder mapping is the inverse of the encoder mapping. The decoding algorithm is not described here but can easily be determined from the encoding algorithm.

The complexity of implementation of the ESVQ is similar to that of the SQV, and as shown in [1], both the storage and the worst-case computational requirements are exponential in \( m \) but only polynomial in \( m \). This makes the scalar-vector quantizers much easier to implement than LBG vector quantizers [7] where the complexity is exponential in \( m \).

In the next section, it is shown that the ESVQ can be used in conjunction with certain block transformations to quantize sources with memory.

V. **Quantization of Sources With Memory**

Quantization of correlated sources using block transforms has been studied in the past and is a popular technique for image and speech coding [8], [9], [15]. The transformations used for this purpose (such as the Karhunen-Loeve transform and the discrete cosine transform) have correlation removing property and for the purpose of quantization, the output vectors of these transformations approximately fit the model of the independent-component vector source described here. The ESVQ is, hence, a good candidate for the efficient quantization of the transformation output vector.

A linear transformation represented by an \( m \times m \) matrix \( A \) can be used to transform a block of \( m \) samples (or an \( m \)-vector) \( \mathbf{x}^T \equiv (x_1, x_2, \ldots, x_m) \) from a stationary correlated source. The transformation applied to \( \mathbf{x} \) results in the output vector \( \mathbf{e} = Ax \). If \( B \) is the inverse transformation matrix, then \( \mathbf{e} = Bc \). The transformation \( A \) is called orthonormal if \( A^{-1} = A^T \). An orthonormal transformation \( A \) corresponds to a rotation of the coordinate system such that: 1) the Euclidean distance between points (vectors) is preserved, i.e., for two vectors \( \mathbf{x}_1, \mathbf{x}_2 \) and the corresponding transformation output vectors \( \mathbf{c}_1, \mathbf{c}_2 \), respectively, \( c_1 - c_2 \rangle^T (c_1 - c_2) = (x_1 - x_2)^T (x_1 - x_2) \), and 2) the sum of the variances is preserved, i.e., \( \sum_{i=1}^{m} \sigma_{i}^2 = \sum_{i=1}^{m} \sigma_{i}^2 \), where \( \sigma_{i}^2 \) and \( \sigma_{i}^2 \) are the variances of the \( i \)-th transform coefficient and the input sample, respectively.

In block transform coding, the transformation output vector \( \mathbf{e} \) is quantized to obtain \( \mathbf{c} \). At the receiver, the inverse transformation \( B \) is performed on \( \mathbf{c} \) to obtain the reconstructed version \( \mathbf{x}' \) of \( \mathbf{x} \). The transformation chosen for this purpose is usually orthonormal and represents the source vector in a special coordinate system—one in which the resulting components are uncorrelated and can, therefore, be efficiently quantized using scalar quantizers. Thus, the decorrelating transformation reduces the redundancy present in the source vector and offers the possibility to use scalar quantizers making the scheme simple to implement. Because of the Euclidean distance preserving nature of these transformations, for the squared-error distortion measure, the reconstruction error of each source vector is the same as the quantization error of the transformed vector and therefore the quantizers can be designed to minimize the expected value of the latter.

It is shown in [15] that for Gaussian input (in the limit of fine quantization) the transform coding gain is realized by an \( m \)-th order orthonormal transformation over direct scalar quantization of the source output is given as

\[
m_{TC} = \frac{1}{N} \sum_{i=1}^{m} \sigma_{i}^2 \cdot \left( \sum_{i=1}^{m} \sigma_{i}^2 \right)^{-\frac{1}{2}}.
\]

The coding gain therefore is the ratio of the arithmetic mean to the geometric mean of the transform coefficient variances.

---

\(^{1}\) The gain here refers to the factor by which the quantization error variance is reduced. The corresponding gain in the signal-to-noise ratio is given as \(10 \log_{10}(m_{TC}) \) in dB.
The optimal \( m \)th-order transformation for stationary Gaussian sources is the Karhunen–Loève transform (KLT) which maximizes the coding gain \( mG_{KLT} \) over all possible linear transformations [8]. The KLT results in a diagonal autocovariance matrix \( R_{cc} \equiv E[CC^T] = AR_{xx}A^T \), where \( R_{xx} \) is the \( m \)th-order symmetric positive-definite source autocovariance matrix. Let \( \lambda_i, i = 1, 2, \ldots, m \), be the \( m \) distinct eigenvalues of \( R_{xx} \) and \( I_i, i = 1, 2, \ldots, m \), be the corresponding eigenvectors scaled such that \( I_i^T I_j = \delta_{ij} \). The KLT matrix has as its \( i \)th row the eigenvector \( I_i \), \( i \in J_m \), of \( R_{xx} \) and the resulting diagonal matrix \( R_{cc} \) has as its diagonal elements the eigenvalues \( \lambda_i, i = 1, 2, \ldots, m \), of \( R_{xx} \). The coding gain \( mG_{KLT} \) of the transformation is given as

\[
mG_{KLT} = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} \lambda_i. \tag{9}
\]

For the KLT, the diagonal nature of the \( R_{cc} \) matrix implies that the transformation output vector has uncorrelated components. This is intuitively satisfying because it says that the optimal transformation is the one that decorrelates the source vector (removing redundancy) before quantizing. For Gaussian samples, uncorrelatedness also implies independence and hence the output vector \( c \) has independent components that can be efficiently quantized using the ESVQ.

Although the KLT is the optimal transformation in terms of coding gain, it is an input-dependent transformation and requires the knowledge of the source autocovariance matrix which may not be available. For such cases, one can use suboptimal source-dependent transformations provided the source is stationary. One such transformation is the discrete cosine transform (DCT) which for sources such as speech and imagery has coding gains very close to that of the KLT [15].

Several researchers have studied block transform coding for the quantization of stationary sources with known correlation properties. Huang and Schultheiss in [8] have applied the KLT to correlated Gaussian sources. The transformation output vector is then quantized using a bank of fixed-length scalar LMQ’s. The rate of each quantizer in the bank is determined optimally by a bit allocation algorithm. The vector components with larger variances receive more bits while some coefficients with small variances may not receive any bits. Farvardin and Lin have studied entropy-constrained quantization of the KLT coefficients for quantizing correlated Gaussian sources [9]. Since optimal entropy-constrained quantizers are hard to design, they have used entropy-coded uniform-threshold quantizers which are known to perform close to the optimum [3]. An optimal entropy allocation algorithm is used to allocate entropy among the various components of the vector. This results in improved performance over the scheme of Huang and Schultheiss, but at the cost of a variable-rate output.

We have studied the fixed-rate quantization of correlated Gaussian sources by using the ESVQ to quantize the KLT transformed vectors. This scheme is hereafter referred to as the KLT-ESVQ scheme. If the source being considered is Gaussian, the equiprobability density surfaces in the \( m \)-dimensional space are (concentric) generalized ellipses [11]. The KLT then corresponds to a rotation of the coordinate system so as to align the axes of these \( m \)-dimensional ellipses. Consider the following ESVQ for quantizing the KLT output vectors for a Gaussian source: 1) the scalar quantizers \( S_1, S_2, \ldots, S_m \) are uniform quantizers with equal step sizes, and 2) the lengths \( \ell_j \) for \( j = 1, 2, \ldots, n_i \) associated with the \( i \)th quantizer \( S_i \) are given by \( \ell_j = w_i \delta_{ij} \), where \( w_i \) is the reciprocal of the variance of the \( i \)th component of the KLT output vector. This ESVQ is similar in spirit to the elliptical vector quantizer in [11] and for an appropriate value of the threshold \( L \), places the codevectors (on a cubic lattice) inside and on the surface of the nearly unit probability (for large \( m \) \( m \)-dimensional generalized ellipse. This is, however, not necessarily the best ESVQ that can be designed for quantizing such sources, especially when \( m \) is not large. An improvement in performance could result if this quantizer is used as the initial choice in the design algorithm of Section III.

The results of the KLT-ESVQ scheme on first-order Gauss–Markov sources with correlation coefficients 0.8 and 0.9 are presented in Tables I and II, respectively. The corresponding ESVQ’s were designed using the algorithm of Section III with lengths effectively quantized to multiples of a quarter of a bit (corresponding to \( b = 4 \)). A total of 50 000 source samples were used in the simulations. The performance results for the Huang and Schultheiss (HS) scheme, Farvardin and Lin (FL) scheme, optimum entropy coded predictive quantizer (ECPQ) [16] (where available) and LBG VQ [7], [17] are also included in the tables for comparison. Due to the design complexity, the LBG VQ results reported here are for a codebook of size 1024 vectors, except at the rate of 3 bits/sample, in which case the codebook size is 512 vectors.

These tables show that, as expected, the performance of the KLT-ESVQ scheme improves with block-length. This is due to two reasons: 1) the performance of the ESVQ itself improves with block-length because the SVQ idea is based on the law of large numbers, and 2) the coding gain of the KLT improves with block-length [15]. The latter is also responsible for the better performance of the HS and FL schemes at larger block lengths. As expected, the KLT-ESVQ scheme for a given encoding rate and block length, performs better than the fixed-rate HS scheme and close to the variable-rate FL scheme. For a fixed block length, the gap between the KLT-ESVQ and the FL scheme is larger at higher rates. This is consistent with the observation in [1] that at higher rates the SVQ requires a larger block-length to perform close to the optimal ECSQ.

From Table I it can be seen that at high rates (fine quantization), the performance of the variable-rate ECPQ is slightly better than the KLT-ESVQ scheme, while at low rates the latter does considerably better. This is because at high rates the ECPQ performs close to (only 1.53 dB below) the rate-distortion limit [16], whereas at low rates it does not perform very well as coarse reproductions are too poor to allow for good predictions. Though for the same block length and rate, vector quantizers perform better than the KLT-ESVQ scheme, the block length of the KLT-ESVQ can be increased to get performance better than LBG VQ at the same rate. For example, in Table II, the KLT-ESVQ scheme at
TABLE I
PERFORMANCE RESULTS (SNR in dB) FOR THE QUANTIZATION OF A FIRST-ORDER GAUSS-MARKOV SOURCE WITH COEFFICIENT OF CORRELATION 0.8

<table>
<thead>
<tr>
<th>Block Length</th>
<th>Rate</th>
<th>1</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>32</th>
<th>ECPQ</th>
<th>LBG VQ</th>
<th>D(R)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>14.62</td>
<td>18.47</td>
<td>19.40</td>
<td>20.00</td>
<td>20.27</td>
<td>20.40</td>
<td>20.96</td>
<td>18.70</td>
<td>22.50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4.40</td>
<td>7.90</td>
<td>8.39</td>
<td>8.69</td>
<td>8.81</td>
<td>8.85</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>14.62</td>
<td>17.78</td>
<td>18.44</td>
<td>18.69</td>
<td>18.77</td>
<td>18.81</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4.58</td>
<td>8.47</td>
<td>9.02</td>
<td>9.30</td>
<td>9.39</td>
<td>9.44</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>16.56</td>
<td>19.84</td>
<td>20.42</td>
<td>20.69</td>
<td>20.78</td>
<td>20.84</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The rate is in bits/sample. Here, HS refers to the Huang-Schultheiss scheme [8] and FL refers to the Farvardin-Lin scheme [9]. The LBG VQ codebook size is 1024 vectors for \( r = 1 \), 2 bits/sample and 512 vectors for \( r = 3 \) bits/sample. The performance of the entropy-coded predictive quantizer (ECPQ) and the source distortion-rate function are also given.

TABLE II
PERFORMANCE RESULTS (SNR in dB) FOR THE QUANTIZATION OF A FIRST-ORDER GAUSS-MARKOV SOURCE WITH COEFFICIENT OF CORRELATION 0.9

<table>
<thead>
<tr>
<th>Block Length</th>
<th>Rate</th>
<th>1</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>24</th>
<th>32</th>
<th>ECPQ</th>
<th>LBG VQ</th>
<th>D(R)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>4.40</td>
<td>10.04</td>
<td>11.21</td>
<td>11.84</td>
<td>12.05</td>
<td>12.17</td>
<td>10.01</td>
<td>11.64</td>
<td>13.23</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>9.30</td>
<td>15.16</td>
<td>16.54</td>
<td>17.23</td>
<td>17.42</td>
<td>17.50</td>
<td>NA</td>
<td>16.27</td>
<td>19.25</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>14.62</td>
<td>20.70</td>
<td>21.77</td>
<td>22.61</td>
<td>22.93</td>
<td>23.11</td>
<td>NA</td>
<td>20.59</td>
<td>25.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4.40</td>
<td>10.02</td>
<td>10.89</td>
<td>11.33</td>
<td>11.52</td>
<td>11.60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4.58</td>
<td>10.60</td>
<td>11.53</td>
<td>11.98</td>
<td>12.13</td>
<td>12.20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10.55</td>
<td>15.98</td>
<td>16.85</td>
<td>17.30</td>
<td>17.46</td>
<td>17.53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>16.56</td>
<td>21.94</td>
<td>22.84</td>
<td>23.30</td>
<td>23.45</td>
<td>23.52</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The rate is in bits/sample. Here, HS refers to the Huang-Schultheiss scheme [8] and FL refers to the Farvardin-Lin scheme [9]. The LBG VQ codebook size is 1024 vectors for \( r = 1 \), 2 bits/sample and 512 vectors for \( r = 3 \) bits/sample. The performance of the entropy-coded predictive quantizer (ECPQ) and the source distortion-rate function are also given.

For Gauss-Markov sources, the gap between the performances of the HS and FL schemes is not much, leaving only a limited room for improvement. The ESVQ-based block transform coding scheme on other sources for which this gap is larger should result in higher gains over the HS scheme. It is well known [15], [18] that the two-dimensional discrete cosine transformation of image data results in close-to-Laplacian distribution of the transform coefficients (except for the “dc” coefficient). Since for Laplacian sources there is a significant gap between the performances of fixed- and variable-length scalar quantizers, the ESVQ offers a potential for improvement over fixed-length quantization of the transform coefficients [19].

VI. ESVQ versus SVQ
So far, we have considered the quantization of independent-component vector sources with the ESVQ. However, these sources can also be quantized using a bank of SVQ's (for scalar memoryless sources) as follows. Since the \( t \)th component of the source \( m \)-vector can be assumed to come from a memoryless scalar subsourse \( X_i, i \in J_m \), an appropriately
designed SVQ can be used to efficiently quantize a block of \( m' \) output samples from this source. Given the block-size of the SVQ for each subsource, its rate-distortion performance can be determined and used for optimal rate allocation among the \( m \) subsources of the vector source. We now compare the ESVQ with this SVQ-based scheme, especially in the context of block transform quantization.

One obvious drawback of the SVQ-based system is the delay it introduces. If \( m'_{\text{max}} \) is the largest of the block lengths of the \( m \) SVQ's, then there is at least an \( m'_{\text{max}} \) vector coding delay introduced into the system. In contrast, the ESVQ introduces no such delay. Furthermore, if the source vector is obtained by using a decorrelating \( m \)-dimensional transformation on a source with memory, the coding delay is \( m \times m'_{\text{max}} \) samples. While this delay may not be of any consequence in still image coding (where the vectors do not form a sequence in time), it can be important in some other applications and will limit the SVQ block length and hence the performance.

In the absence of any constraints on delay or complexity, the asymptotic performances (for large block lengths) of the ESVQ- and SVQ-based block transform quantization schemes are expected to be the same. However, as explained next, for practical systems that place a limit on the complexity, the ESVQ-based system has an edge. In many practical schemes that use the KLT or DCT, the low-frequency components (subsources) of the transform vectors have variances significantly larger than those of the high frequency components. Thus, some subsources must be encoded with significantly higher rates than the average rate. Since the complexity of implementation of the SVQ is exponential in rate, the block length of SVQ's designed for these sources is usually small and limited by the allowed complexity. This is particularly undesirable because at a high rate the SVQ requires a larger block-length to perform close to the optimal variable-length encoded scalar quantizer [1]. For the ESVQ, on the other hand, the larger variance subsources are quantized together with the smaller variance sources allowing larger block-lengths (than the SVQ) for a similar complexity. This means that for the ESVQ, even the large variance subsources can enjoy the benefits of a large block length. For a given limit on complexity, in our experience, the ESVQ-based scheme usually outperforms the SVQ-based block transform coding scheme.

Table III compares the performance results (SNR in dB) of the 32-dimensional KLT-ESVQ scheme to the corresponding results of the SVQ-based system for a Gauss–Markov source with coefficient of correlation 0.9. For these results, the maximum SVQ dimension was taken to be \( m'_{\text{max}} = 32 \). Also, to place a limit on the complexity, the total rate of the SVQ was limited to 96 bits/block. Under the above constraints, SVQ's with maximum possible block-lengths were designed. This table confirms the claim that for comparable complexities, the ESVQ-based block transform quantizer performs better than the SVQ-based scheme; the ESVQ also introduces a significantly smaller coding delay.

### VII. Summary and Conclusion

This paper extends the fixed-rate scalar-vector quantizer to vector sources for which each component is a stationary memoryless scalar subsource independent of the other components. A suboptimal design algorithm for the ESVQ was presented along with fast algorithms for codebook search and codevector encoding. It is shown that the ESVQ can be applied to the quantization of stationary sources with memory. This is done by first using the KLT or DCT to decorrelate a block of source data. The transform output vector is then quantized using the SVQ. Numerical results were presented for first-order Gauss–Markov sources and it was shown that the KLT-ESVQ scheme improves upon the performance of the fixed-rate HS scheme and performs close to the variable-rate FL scheme. It was also shown that the KLT-ESVQ scheme performs better than implementable vector quantizers (especially at high rates). Because of the large gap in the performances of fixed- and variable-length scalar quantizers for Laplacian sources, the suboptimal scheme that uses the ESVQ to quantize the DCT output vectors can be used for the efficient quantization if image data [19].

### REFERENCES


