Vector Quantization

Robert M. Gray

A vector quantizer is a system for mapping a sequence of continuous or discrete vectors into a digital sequence suitable for communication over or storage in a digital channel. The goal of such a system is data compression: to reduce the bit rate so as to minimize communication channel capacity or digital storage memory requirements while maintaining the necessary fidelity of the data. The mapping for each vector may or may not have memory in the sense of depending on past actions of the coder, just as in well established scalar techniques such as PCM, which has no memory, and predictive quantization, which does. Even though information theory implies that one can always obtain better performance by coding vectors instead of scalars, scalar quantizers have remained by far the most common data compression system because of their simplicity and good performance when the communication rate is sufficiently large. In addition, relatively few design techniques have existed for vector quantizers.

During the past few years several design algorithms have been developed for a variety of vector quantizers and the performance of these codes has been studied for speech waveforms, speech linear predictive parameter vectors, images, and several simulated random processes. It is the purpose of this article to survey some of these design techniques and their applications.

DATA compression is the conversion of a stream of analog or very high rate discrete data into a stream of relatively low rate data for communication over a digital communication link or storage in a digital memory. As digital communication and secure communication have become increasingly important, the theory and practice of data compression have received increased attention. While it is true that in many systems bandwidth is relatively inexpensive, e.g., fiber optic and cable TV links, in most systems the growing amount of information that users wish to communicate or store necessitates some form of compression for efficient, secure, and reliable use of the communication or storage medium.

A prime example arises with image data, where simple schemes require bit rates too large for many communication links or storage devices. Another example where compression is required results from the fact that if speech is digitized using a simple PCM system consisting of a sampler followed by scalar quantization, the resulting signal will no longer have a small enough bandwidth to fit on ordinary telephone channels. That is, digitization (which may be desirable for security or reliability) causes bandwidth expansion. Hence data compression will be required if the original communication channel is to be used.

The two examples of image compression and speech compression or, as they are often called, image coding and speech coding, are probably the currently most important applications of data compression. They are also among the most interesting for study because experience has shown that both types of data exhibit sufficient structure to permit considerable compression with sufficiently sophisticated codes.

Such conversion of relatively high rate data to lower rate data virtually always entails a loss of fidelity or an increase in distortion. Hence a fundamental goal of data compression is to obtain the best possible fidelity for the given rate or, equivalently, to minimize the rate required for a given fidelity. If a system has a sufficiently high rate constraint, then good fidelity is relatively easy to achieve and techniques such as PCM, transform coding, predictive coding, and adaptive versions of these techniques have become quite popular because of their simplicity and good performance [1, 2, 3]. All of these techniques share a fundamental property: The actual quantization or coding or conversion of continuous quantities into discrete quantities is done on scalars, e.g., on individual real-valued samples of waveforms or pixels of images. PCM does this in a memoryless fashion; that is, each successive input is encoded using a rule that does not depend on any past inputs or outputs of the encoder. Transform coding does it by first taking block transforms of a vector and then scalar coding the coordinates of the transformed vector. Predictive coding does it by quantizing an error term formed as the difference between the new sample and a prediction of the new sample based on past coded outputs.

A fundamental result of Shannon's rate-distortion theory, the branch of information theory devoted to data compression, is that better performance can always be achieved by coding vectors instead of scalars, even if the data source is memoryless, e.g., consists of a sequence of independent random variables, or if the data compression system can have memory, i.e., the action of an encoder at each time is permitted to depend on past encoder inputs or outputs [4, 5, 6, 7, 8]. While some traditional compression schemes such as transform coding operate on vectors and achieve significant improvement over PCM, the quantization is still accomplished on scalars and hence these systems are, in a Shannon sense, inherently suboptimal: better performance is always achievable *in theory* by coding vectors instead of scalars, even if the scalars have been produced by preprocessing the original input data so as to make them uncorrelated or independent!

0740-7467/84/0400-0004$1.00©1984 IEEE

This theory had a limited impact on actual system design because 1) the Shannon theory does not provide constructive design techniques for vector coders, and 2) traditional scalar coders often yield satisfactory performance with enough adaptation and fine tuning. As a result, few design techniques for vector quantizers were considered in the literature prior to the late 1970's when it was found that a simple algorithm of Lloyd [9] for the iterative design of scalar quantization or PCM systems extended in a straightforward way to the design of memoryless vector quantizers, that is, of vector quantizers which encode successive input vectors in a manner not depending on previous encoder input vectors or their coded outputs. Variations of the basic algorithm have since proved useful for the design of vector quantizers with and without memory for a variety of data sources including speech waveforms, speech parameter vectors, images, and several random process models, the latter being useful for gauging the performance of the resulting codes with the optimal performance bounds of information theory.

This paper is intended as a survey of the basic design algorithm and many of its variations and applications. We begin with the simplest example of a memoryless vector quantizer, a vector generalization of PCM. For convenience we use the shorthand VQ for both vector quantization and vector quantizer. Necessary properties of optimal quantizers are described and an algorithm given which uses these properties to iteratively improve a code. For concreteness, we focus on two examples of distortion measures: the ubiquitous mean-squared error and the Itakura-Saito distortion. The first example, which is popular in waveform coding applications, provides a geometric flavor to the development; the second example, which is useful in voice coding applications, helps to demonstrate the generality and power of the technique.

Next, various techniques are described for designing the initial codes required by the algorithm. These techniques also indicate some useful structure that can be imposed on vector quantizers to make them more implementable. Several variations of the basic VQ are described which permit reduced complexity or memory or both at the expense of a hopefully tolerable loss of performance. These include tree-searched codes, product codes, and multistep codes.

We then turn from memoryless vector quantizers to those with memory: feedback vector quantizers such as vector predictive quantizers and finite-state vector quantizers. These codes are not yet well understood, but they possess a structure highly suited to VLSI implementation and initial studies suggest that they offer significant performance gains.

For comparison, we also briefly describe trellis encoding systems or "lookahead" or "delayed decision" or "multipath search" codes which use the same decoder as a feedback vector quantizer but which permit the encoder to base its decision on a longer input data sequence.

A final general code structure is described which uses vector quantization to adapt a waveform coder, which may be another VQ.

We next present a variety of simulation results describing the performance of various VQ systems on various data sources. Examples of all of the above VQ varieties are tested for waveform coding applications on two common data sources: a Gauss Markov source and real sampled speech. One bit per sample coders for these sources are compared on the basis of performance, memory requirements, and computational complexity. Both memoryless and simple feedback vector quantizers are studied for voice coding applications at a rate of 0.062 bits/sample and less and for image coding at a rate of 0.5 bit per sample. One example is given of a simple adaptive predictive vector quantizer for speech waveform coding.

By studying a variety of coding systems on common data sources, the results yield some general comparisons and trends among the various vector quantization techniques. The reader should, however, keep two caveats in mind when interpreting such quantitative results: First, the emphasis here is on low bit rate systems, e.g., speech coders using 1 bit per sample or less and image coders ½ bit per pixel. Comparisons favoring certain systems at such low rates may not be valid for the same systems at higher rates. Second, the numbers reported here are intended to provide comparisons for different systems used on common data sources; they can be compared with other numbers reported in the literature only with great care: the input data and the system design parameters such as sampling rate and pre- or post-filtering may be quite different.

Applications of vector quantization to real data sources such as sampled speech waveforms and images are still young and the algorithms do not yet incorporate the sophisticated "bells and whistles" of many well-established scalar quantization schemes. The preliminary experiments described here, using fairly simple vector quantizers with and without memory, demonstrate that the general approach holds considerable promise for some applications. For example, good quality vocoding systems using VQ and the Itakura-Saito distortion have been developed at 800 bits per second, a significant reduction in the bit rate previously required for comparable quality [10]. While the compression achieved so far in waveform coding and image coding applications using the squared-error distortion has not yet been as significant, we believe that it has yielded comparable or better performance at low rates than traditional scalar schemes of greater complexity. The quality of the ½ bit per pixel images shown here is promising given the simplicity of the coding scheme used.

We attempt to use the minimum of mathematics and a maximum of English in the presentation so as to focus on the intuitive ideas underlying the design and operation of vector quantizers. The detailed descriptions of the various algorithms can be found in the cited references. The reader is also referred to a recent tutorial by Gersho and Cuperman [11] which presents a brief overview of VQ applied to speech waveform coding.

MEMORYLESS VECTOR QUANTIZERS

In this section we introduce the basic definition of memoryless vector quantizers, their properties, and an algorithm for their design.

Quantization

Mathematically, a k-dimensional memoryless vector quantizer or, simply, a VQ (without modifying adjectives) consists of two mappings: an encoder γ which assigns to each input vector $\mathbf{x} = (x_0, x_1, \cdots, x_{k-1})$ a channel symbol $\gamma(\mathbf{x})$ in some channel symbol set $\mathbf{M}$, and a decoder β assigning to each channel symbol v in $\mathbf{M}$ a value in a reproduction alphabet $\hat{\mathbf{A}}$. The channel symbol set is often assumed to be a space of binary vectors for convenience, e.g., $\mathbf{M}$ may be the set of all 2^R binary R-dimensional vectors. The reproduction alphabet may or may not be the same as the input vector space; in particular, it may consist of real vectors of a different dimension.

If $\mathbf{M}$ has M elements, then the quantity $R = \log_2 M$ is called the *rate* of the quantizer in bits per vector and $r = R/k$ is the rate in bits per symbol or, when the input is a sampled waveform, bits per sample.

The application of a quantizer to data compression is depicted in the standard Fig. 1. The input data vectors might be consecutive samples of a waveform, consecutive parameter vectors in a voice coding system, or consecutive rasters or subrasters in an image coding system. For integer values of R it is useful to think of the channel symbols, the encoded input vectors, as binary R-dimensional vectors. As is commonly done in information and communication theory, we assume that the channel is noiseless, that is, that $U_n = \hat{U}_n$. While real channels are rarely noiseless, the joint source and channel coding theorem of information theory implies that a good data compression system designed for a noiseless channel can be combined with a good error correction coding system for a noisy channel in order to produce a complete system. In other words, the assumption of a noiseless channel is made simply to focus on the problem of data compression system design and not to reflect any practical model.

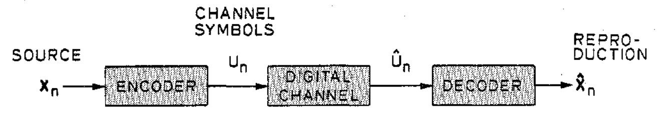


Figure 1. Data Compression System. The data or information source $\{X_n;\ n = 0, 1, \ldots\}$ is a sequence of random vectors. The encoder produces a sequence of channel symbols $\{U_n;\ n = 0, 1, 2, \ldots\}$. The sequence $\{\hat{U}_n;\ n = 0, 1, 2, \ldots\}$ is delivered to the receiver by the digital channel. The decoder then maps this sequence into the final reproduction sequence of vectors $\{\hat{X}_n;\ n = 0, 1, 2, \ldots\}$.

Observe that unlike scalar quantization, general VQ permits fractional rates in bits per sample. For example, scalar PCM must have a bit rate of at least 1 bit per sample while a k dimensional VQ can have a bit rate of only $1/k$ bits per sample by having only a single binary channel symbol for k-dimensional input vectors.

The goal of such a quantization system is to produce the "best" possible reproduction sequence for a given rate R. To quantify this idea, to define the performance of a quantizer, and to complete the definition of a quantizer, we require the idea of a distortion measure.

Distortion

A distortion measure d is an assignment of a cost $d(\mathbf{x}, \hat{\mathbf{x}})$ of reproducing any input vector $\mathbf{x}$ as a reproduction vector $\hat{\mathbf{x}}$. Given such a distortion measure, we can quantify the performance of a system by an average distortion $Ed(\mathbf{X}, \hat{\mathbf{X}})$ between the input and the final reproduction: A system will be good if it yields a small average distortion. In practice, the important average is the long term sample average or time average

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} d(\mathbf{X}_i, \hat{\mathbf{X}}_i) \tag{1}$$

provided, of course, that the limit makes sense. If the vector process is stationary and ergodic, then, with probability one, the limit exists and equals an expectation $E(d(\mathbf{X}, \hat{\mathbf{X}}))$. For the moment we will assume that such conditions are met and that such long term sample averages are given by expectations. Later remarks will focus on the general assumptions required and their implications for practice.

Ideally a distortion measure should be tractable to permit analysis, computable so that it can be evaluated in real time and used in minimum distortion systems, and subjectively meaningful so that large or small quantitative distortion measures correlate with bad and good subjective quality. Here we do not consider the difficult and controversial issues of selecting a distortion measure; we assume that one has been selected and consider means of designing systems which yield small average distortion. For simplicity and to ease exposition, we focus on two important specific examples:

(1) *The squared error distortion measure:* Here the input and reproduction spaces are k-dimensional Euclidean space

$$d(\mathbf{x}, \hat{\mathbf{x}}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2 = \sum_{i=0}^{k-1} (x_i - \hat{x}_i)^2,$$

the square of the Euclidean distance between the vectors. This is the simplest distortion measure and the most common for waveform coding. While not subjectively meaningful in many cases, generalizations permitting input-dependent weightings have proved useful and only slightly more complicated. For the squared-error distortion it is common practice to measure the performance of a system by the signal-to-noise ratio (or signal-to-quantization-noise ratio)

$$SNR = 10 \log_{10} \frac{E(\|\mathbf{X}\|^2)}{E[d(\mathbf{X}, \hat{\mathbf{X}})]}.$$

This corresponds to normalizing the average distortion by the average energy and plotting it on a logarithmic scale: Large (small) SNR corresponds to small (large) average distortion.

(2) *The (modified) Itakura-Saito distortion:* This distortion measure is useful in voice coding applications where the receiver is sent a linear model of the underlying voice production process. The distortion measure is based on the "error matching measure" developed in the pioneering work of Itakura and Saito on the PARCOR or LPC approach to voice coding [12]. More generally, this distortion measure is a special case of a minimum relative entropy or discrimination measure; VQ using such distortion measures can be viewed as an application of the minimum relative entropy pattern classification technique introduced by Kullback [13] as an application of information theory to statistical pattern classification. (See also [14, 15].)

We here introduce a minimum of notation to present a definition of the Itakura-Saito distortion measure. Details and generalizations may be found in [16, 17, 14, 15]. Here the input vector can again be considered as a collection of consecutive waveform samples. Now, however, the output vectors have the form $\hat{\mathbf{x}} = (\alpha, a_1, a_2, \cdots, a_p)$, where α is a positive gain or residual energy term and where the a_i with $a_0 = 1$ are inverse filter coefficients in the sense that if

$$A(z) = \sum_{i=0}^{p} a_i z^{-i}$$

then the all-pole filter with z-transform $1/A(z)$ is a stable filter. Here the reproduction vectors may be thought of as all-pole models for synthesizing the reproduction at the receiver using a locally generated noise or periodic source, in other words, as the filter portion of a linear predictive coding (LPC) model in a vocoding (voice coding) system. The Itakura-Saito distortion between the input vector and the model can be defined in the time domain as

$$d(\mathbf{x}, \hat{\mathbf{x}}) = \frac{\mathbf{a}^t \mathbf{R}(\mathbf{x}) \mathbf{a}}{\alpha} - \ln \frac{\alpha_p(\mathbf{x})}{\alpha} - 1,$$

where $\mathbf{a}^t = (1, a_1, \cdots, a_p)$, $\mathbf{R}(\mathbf{x})$ is the $(p+1) \times (p+1)$ sample autocorrelation matrix of the input vector $\mathbf{x}$, and where $\alpha_p(\mathbf{x})$ is an input gain (residual energy) term defined as the minimum value of $\mathbf{b}^t\mathbf{R}(\mathbf{x})\mathbf{b}$, where the minimum is taken over all vectors $\mathbf{b}$ with first component equal to 1. There are many equivalent forms of the distortion measure, some useful for theory and some for computation. Frequency domain forms show that minimizing the above distortion can be interpreted as trying to match the sample spectrum of the input vector to the power spectral density of the linear all-pole model formed by driving the filter with z-transform $1/A(z)$ by white noise with constant power spectral density $\sqrt{\alpha}$.

The above formula for the distortion is one of the simplest, yet it demonstrates that the distortion measure is indeed complicated—it is not a simple function of an error vector, it is not symmetric in its input and output arguments, and it is not a metric or distance. Because of the intimate connection of this distortion measure with LPC vocoding techniques, we will refer to VQ's designed using this distortion measure as LPC VQ's.

Average distortion

As the average distortion quantifies the performance of a system and since we will be trying to minimize this quantity using good codes, we pause to consider what the average means in theory and in practice.

As previously noted, in practice it is the long term sample average of (1) that we actually measure and which we would like to be small. If the process is stationary and ergodic, then this limiting time average is the same as the mathematical expectation. The mathematical expectation is useful for developing information theoretic performance bounds, but it is often impossible to calculate in practice because the required probability distributions are not known, e.g., there are no noncontroversial generally accepted accurate probability distributions for real speech and image data. Hence a pragmatic approach to system design is to take long sequences of training data, estimate the "true" but unknown expected distortion by the sample average, and attempt to design a code that minimizes the sample average distortion for the training sequence. If the input source is indeed stationary and ergodic, the resulting sample average should be nearly the expected value and the same code used on future data should yield approximately the same averages [18].

The above motivates a training sequence based design for stationary and ergodic data sources. In fact, even if the "true" probability distributions are known as in the case of a Gauss Markov source, the training sequence approach reduces to a standard Monte Carlo approach.

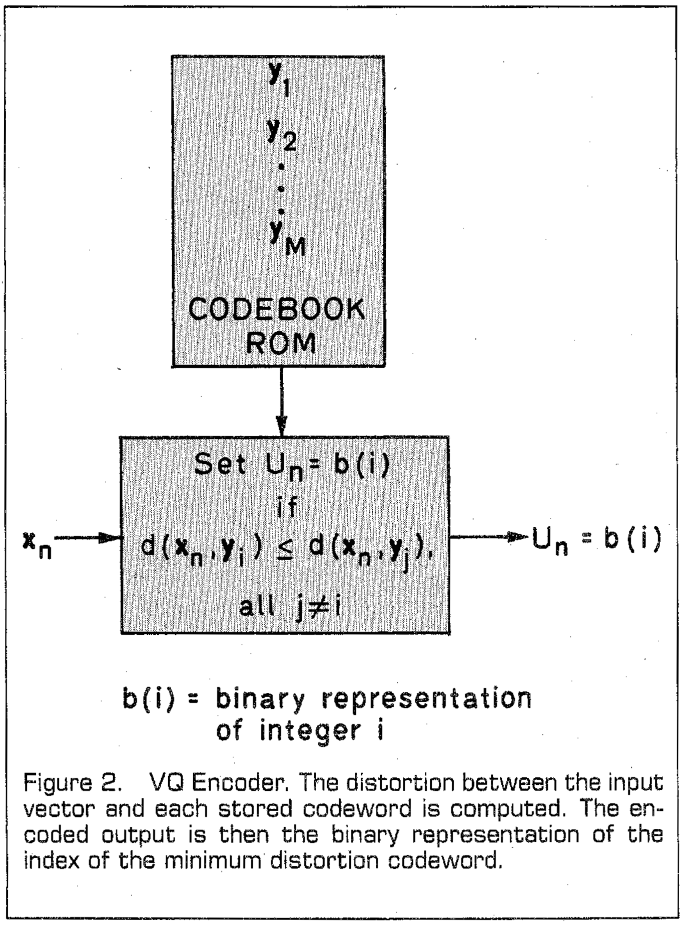
An immediate objection to the above approach, however, is whether or not it makes sense for real sources which may be neither stationary nor ergodic. The answer is an emphatic "yes" in the following sense: The desired property is that if we design a code based on a *sufficiently long* training sequence and then use the code on future data produced by the same source, then the performance of the code on the new data should be roughly that achieved on the training data. The theoretical issue is to provide conditions under which this statement can be made rigorous. For reasonable distortion measures, a sufficient condition for this to be true for memoryless VQ design is that the source be asymptotically mean stationary, it need not be either stationary nor ergodic [19, 20, 21, 22, 23]. Asymptotically mean stationary sources include all stationary sources, block (or cyclo) stationary sources, and asymptotically stationary sources. Processes such as speech which exhibit distinct short term and long term stationarity properties are well modeled by asymptotically mean stationary sources [21].

The key point here is that the general design approach using long training sequences does not require either ergodicity nor stationarity to have a solid mathematical foundation. In fact, the mathematics suggest the following pragmatic approach: Try to design a code which minimizes the sample average distortion for a very long training sequence. Then use the code on test sequences produced by the same source, but not in the training sequence. If the performance is reasonably close to the design values, then one can have a certain amount of confidence that the code will continue to yield roughly the same performance in the future. If the training and test performance are significantly different, then probably the training sequence is not sufficiently long. In other words, do not try to prove mathematically that a source is asymptotically mean stationary, instead try to design codes for it and then see if they work on new data.

Henceforth for brevity we will write expectations with the assumption that they are to be interpreted as shorthand for long term sample averages. (A sample average $L^{-1}\sum_{i=0}^{L-1} d(\mathbf{X}_i,\hat{\mathbf{X}}_i)$ is, in fact, an expectation with respect to the sample distribution which assigns a probability of $1/L$ to each vector in the training sequence.)

Properties of optimal quantizers

A VQ is optimal if it minimizes an average distortion $Ed\{\mathbf{X},\beta[\gamma(\mathbf{X})]\}$. Two necessary conditions for a VQ to be optimal follow easily using the same logic as in Lloyd's [9] classical development for optimal PCM with a mean-squared error distortion measure. The following definition is useful for stating these properties: The collection of possible reproduction vectors $\mathbf{C} = \{\text{all } \mathbf{y} : \mathbf{y} = \beta(v), \text{ some } v \text{ in } \mathbf{M}\}$ is called the *reproduction codebook* or, simply, *codebook* of the quantizer and its members called *codewords* (or templates). The encoder knows the structure of the decoder and hence all of the possible final output codewords.



$\mathbf{y}_1$
$\mathbf{y}_2$
$\mathbf{y}_M$
CODEBOOK ROM
Set $U_n = b(i)$ if $d(\mathbf{x}_n,\mathbf{y}_i) \leq d(\mathbf{x}_n,\mathbf{y}_j)$, all $j \neq i$
$\mathbf{x}_n$
$U_n = b(i)$
b(i) = binary representation of integer i

Figure 2. VQ Encoder. The distortion between the input vector and each stored codeword is computed. The encoded output is then the binary representation of the index of the minimum distortion codeword.

Property 1: Given the goal of minimizing the average distortion and given a specific decoder β, no memoryless quantizer encoder can do better than select the codeword v in $\mathbf{M}$ that will yield the minimum possible distortion at the output, that is, to select the channel symbol v yielding the minimum

$$d\{\mathbf{x},\beta[\gamma(\mathbf{x})]\} = \min_{v\in\mathbf{M}} d[\mathbf{x},\beta(v)] = \min_{y\in C} d(\mathbf{x},y). \quad (2)$$

That is, for a given decoder in a memoryless vector quantizer the best encoder is a minimum distortion or nearest neighbor mapping

$$\gamma(\mathbf{x}) = \min_{v\in\mathbf{M}}{}^{-1} d[\mathbf{x},\beta(v)], \quad (3)$$

where the inverse minimum notation means that we select the v giving the minimum of (2).

Gersho [24] calls a quantizer with a minimum distortion encoder a Voronoi quantizer since the Voronoi regions about a set of points in a space correspond to a partition of that space according to the nearest-neighbor rule. The word quantizer, however, is practically always associated with such a minimum distortion mapping. We observe that such a vector quantizer with such a minimum distortion encoder is exactly the Shannon model for a block source code subject to a fidelity criterion which is used in information theory to develop optimal performance bounds for data compression systems.

An encoder γ can be thought of as a partition of the input space into cells where all input vectors yielding a common reproduction are grouped together. Such a partition according to a minimum distortion rule is called a Voronoi or Dirichlet partition. A general minimum distance VQ encoder is depicted In Fig. 2.

A simple example of such a partition and hence of an encoder is depicted in Fig. 3 (a more interesting example follows shortly). Observe that this vector quantizer is just two uses of a scalar quantizer in disguise.

As the minimum distortion rule optimizes the encoder of a memoryless VQ for a decoder, we can also optimize the decoder for a given encoder.

Property 2: Given an encoder γ, then no decoder can do better than that which assigns to each channel symbol v the generalized centroid (or center of gravity or barycenter) of all source vectors encoded into v, that is,

$$\beta(v) = cent(v) = \min_{\hat{\mathbf{x}}\in\hat{A}}{}^{-1} E(d(\mathbf{X},\hat{\mathbf{x}}) \mid \gamma(\mathbf{X}) = v), \quad (4)$$

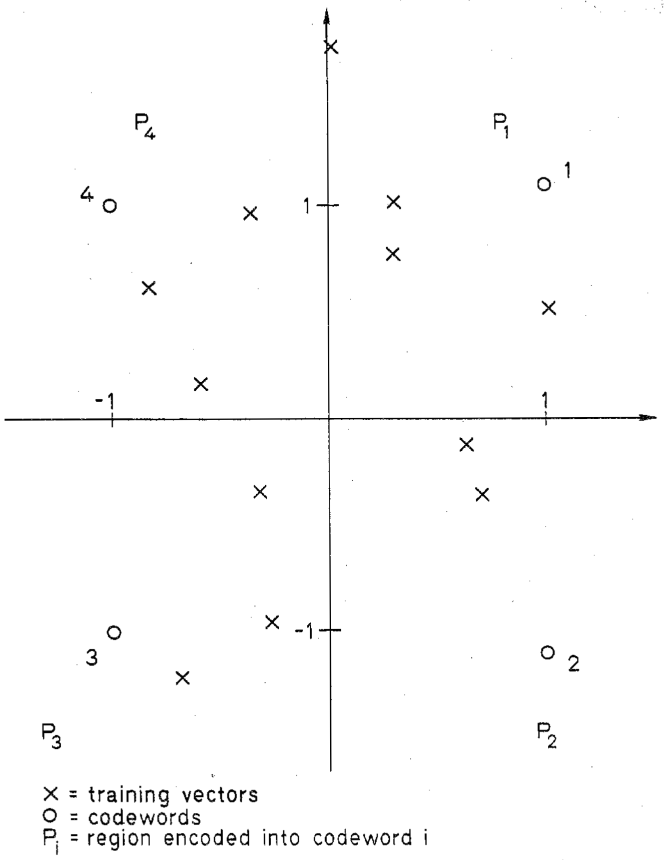


Figure 3. Two-Dimensional Minimum Distortion Partition. The four circles are the codewords of a two-dimensional codebook. The Voronoi regions are the quadrants containing the circles. The x's were produced by a training sequence of twelve two-dimensional Gaussian vectors. Each input vector is mapped into the nearest-neighbor codeword, that is, the circle in the same quadrant.

that is, $\beta(v)$ is the vector yielding the minimum conditional average distortion given that the input vector was mapped into v.

While minimizing such a conditional average may be quite difficult for an arbitrary random process and distortion measure, it is often easy to find for a sample distribution and a nice distortion measure. For example, the centroid in the case of a sample distribution and a squared-error distortion measure is simply the ordinary Euclidean centroid or the vector sum of all input vectors encoded into the given channel symbol, that is, given the sample distribution defined by a training sequence $\{\mathbf{x}_i;\ i = 0, 1, \ldots, L-1\}$, then

$$cent(v) = \frac{1}{i(v)} \sum_{\mathbf{x}_i : \gamma(\mathbf{x}_i) = v} x_i,$$

where $i(v)$ is the number of indices i for which $\gamma(\mathbf{x}_i) = v$. The Euclidean centroids of the example of Fig. 3 are depicted in Fig. 4. (The numerical values may be found in [25].) The new codewords better represent the training vectors mapping into the old codewords, but they yield a different minimum distortion partition of the input alphabet, as indicated by the broken line in Fig. 3. This is the key of the algorithm: iteratively optimize the codebook for the old encoder and then use a minimum distortion encoder for the new codebook.

The Itakura-Saito distortion example is somewhat more complicated, but still easily computable. As with the squared error distortion, one groups all input vectors yielding a common channel symbol. Instead of averaging the vectors, however, the sample autocorrelation matrices for all of the vectors are averaged. The centroid is then given by the standard LPC all-pole model for this average autocorrelation, that is, the centroid is found by a standard Levinson's recursion run on the average autocorrelation.

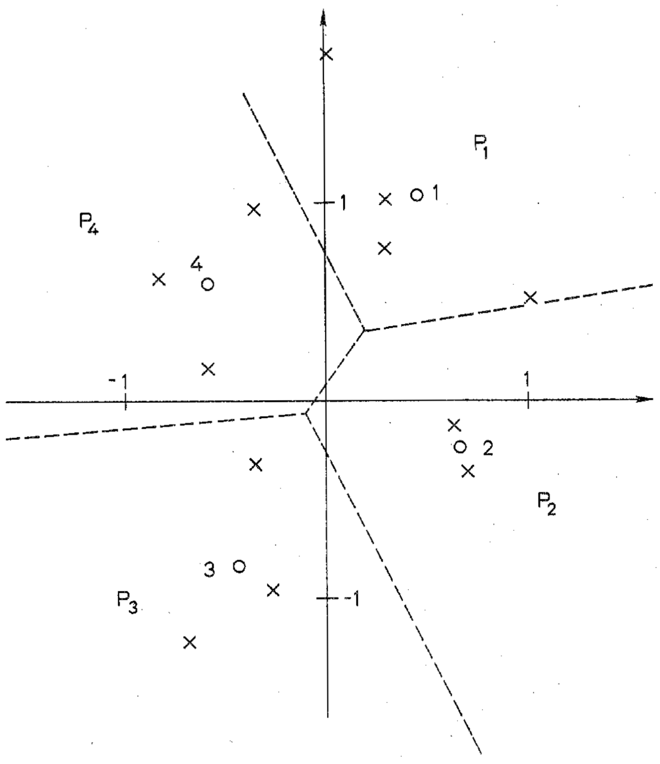


Figure 4. Centroids of Figure 3. The new centroids of the old Voronoi regions of Fig. 3 are drawn as circles. Note that the centroid computation has moved the codewords to better represent the input vectors which yielded those codewords, that is, if one used the same encoder (as in Fig. 3), but replaced the reproduction codewords produced at the decoder by these new centroids, the average distortion would decrease. The broken line delineates the new Voronoi regions for these codewords.

The generalized Lloyd algorithm

The fact that the encoder can be optimized for the decoder and vice versa formed the basis of Lloyd's original optimal PCM design algorithm for a scalar random variable with a known probability density function and a squared error distortion. The general VQ design algorithms considered here are based on the simple observation that Lloyd's basic development is valid for vectors, for sample distributions, and for a variety of distortion measures. The only requirement on the distortion measure is that one can compute the centroids. The basic algorithm is the following:

Step 0. Given: A training sequence and an initial decoder.
Step 1. Encode the training sequence into a sequence of channel symbols using the given decoder minimum distortion rule. If the average distortion is small enough, quit.
Step 2. Replace the old reproduction codeword of the decoder for each channel symbol v by the centroid of all training vectors which mapped into v in Step 1. Go to Step 1.

Means of generating initial decoders will be considered in the next section. Each step of the algorithm must either reduce average distortion or leave it unchanged. The algorithm is usually stopped when the relative distortion decrease falls below some small threshold. The algorithm was developed for vector quantizers, training sequences, and general distortion measures by Linde, Buzo, and Gray [25] and it is sometimes referred to as the LBG algorithm. Previously Lloyd's algorithm had been considered for vectors and difference distortion measures in cluster analysis and pattern recognition problems (e.g., MacQueen [26] and Diday and Simon [27]) and in two-dimensional quantization (e.g., Chen [28] and Adoul *et al.* [29]). Only recently, however, has it been extensively studied for vector quantization applications using several different distortion measures.

Before continuing, it should be emphasized that such iterative improvement algorithms need not in general yield truly optimum codes. It is known that subject to some mathematical conditions the algorithm will yield locally optimum quantizers, but in general there may be numerous such codes and many may yield poor performance. (See, e.g., [30].) It is often useful, therefore, to enhance the algorithm's potential by providing it with good initial codebooks and perhaps by trying it on several different initial codebooks.

INITIAL CODEBOOKS

The basic design algorithm of the previous section is an iterative improvement algorithm and requires an initial code to improve. Two basic approaches have been developed: One can start with some simple codebook of the correct size or one can start with a simple small codebook and recursively construct larger ones.

"Random" codes

Perhaps the simplest example of the first technique is that used in the k-means variation of the algorithm [26]: Use the first 2^R vectors in the training sequence as the initial codebook. An obvious modification more natural for highly correlated data is to select several widely spaced words from the training sequence. This approach is sometimes called random code generation, but we avoid this nomenclature because of its confusion with the random code techniques of information theory which are used to prove the performance bounds.

Product codes

Another example of the first approach is to use a scalar code such as a uniform quantizer k times in succession and then prune the resulting vector codebook down to the correct size. The mathematical model for such a code is a product code, which we pause to define for current and later use: Say we have a collection of codebooks C_i, $i = 0, 1, \ldots, m-1$, each consisting of M_i vectors of dimension k_i and having rate $R_i = \log_2 M_i$ bits per vector. Then the *product codebook* C is defined as the collection of all $M = \Pi_i M_i$ possible concatenations of m words drawn successively from the m codebooks C_i. The dimension of the product codebook is $k = \Sigma_{i=0}^{m-1} k_i$, the sum of the dimensions of the component codebooks. The product code is denoted mathematically as a Cartesian product:

$$C = \mathop{\times}_{i=0}^{m-1} C_i = \{\textit{all vectors of the form } (\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_{m-1});\ \hat{\mathbf{x}}_i \textit{ in } C_i;\ i = 0, 1, \ldots, m-1\}$$

Thus, for example, using a scalar quantizer with rate R/k k times in succession yields a product k-dimensional vector quantizer of rate R bits per vector. This product code can be used as an initial code for the design algorithm. The scalar quantizers may be identical uniform quantizers with a range selected to match the source, or they may be different, e.g., a positive codebook for a gain and uniform quantizers for $[-1,1]$ for reflection coefficients in an LPC VQ system.

In waveform coding applications where the reproduction and input alphabets are the same—k-dimensional Euclidean space—an alternative product code provides a means of growing better initial guesses from smaller dimensional codes [31]. Begin with a scalar quantizer C_0 and use a two-dimensional product code $C_0 \times C_0$ as an initial guess for designing a two-dimensional VQ. On completion of the design we have a two-dimensional code, say C^2. Form an initial guess for a three dimensional code as all possible pairs from C^2 and scalars from C_0, that is, use the product code $C^2 \times C_0$ as an initial guess. Continuing in this way, given a good $k-1$ dimensional VQ described by a codebook C^{k-1}, an initial guess for a k-dimensional code design is the product code $C^{k-1} \times C_0$. One can also use such product code constructions with a different initial scalar code C_0, such as those produced by the scalar version of the next algorithm.

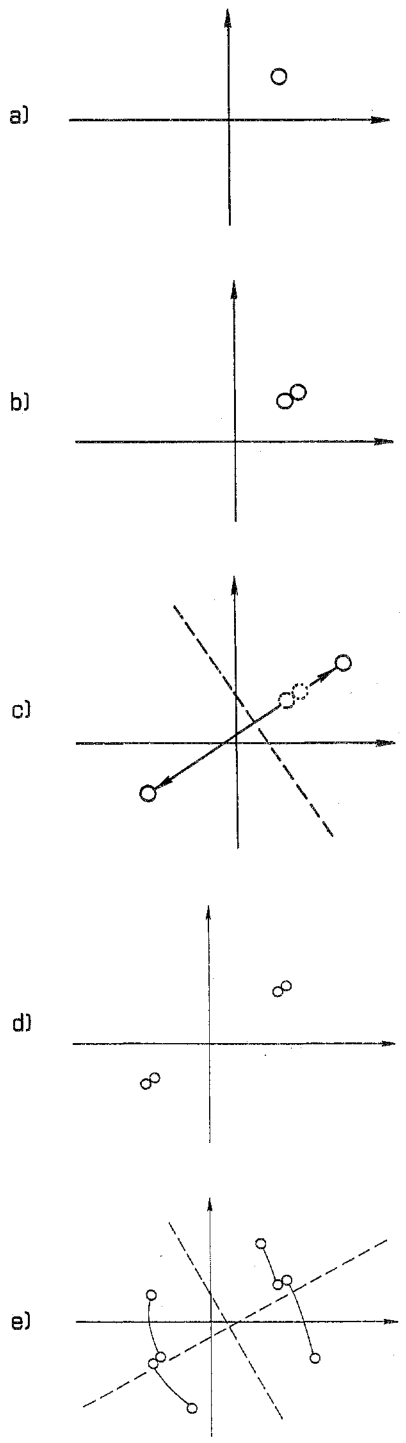


Figure 5. Splitting. A large code is defined in stages: at each stage each codeword of a small code is split into two new codewords, giving an initial codebook of twice the size. The algorithm is run to get a new better codebook. (a) Rate 0: The centroid of the entire training sequence. (b) Initial Rate 1: The one codeword is split to form an initial guess for a two word code. (c) Final Rate 1: The algorithm produces a good code with two words. The dotted line indicates the Voronoi regions. (d) Initial Rate 2: The two words are split to form an initial guess for a four word code. (e) Final Rate 2: The algorithm is run to produce a final four word code.

Splitting

Instead of constructing long codes from smaller dimensional codes, we can construct a sequence of bigger codes having a fixed dimension using a "splitting" technique [25,16]. This method can be used for any fixed dimension, including scalar codes. Here one first finds the optimum 0 rate code—the centroid of the entire training sequence, as depicted in Fig. 5a for a two-dimensional input alphabet. This single codeword is then split to form two codewords (Fig. 5b). For example, the energy can be perturbed slightly to form a second distinct word or one might purposefully find a word distant from the first. It is convenient to have the original codeword a member of the new pair to ensure that the distortion will not increase. The algorithm is then run to get a good rate 1 bit per vector code as indicated in Fig. 5c. The design continues in this way in stages as shown: the final code of one stage is split to form an initial code for the next.

VARIATIONS OF MEMORYLESS VECTOR QUANTIZERS

In this section we consider some of the variations of memoryless vector quantization aimed at reducing the computation or memory requirements of a full search memoryless VQ.

Tree-searched VQ

Tree-searched vector quantizers were first proposed by Buzo *et al.* [16] and are a natural byproduct of the splitting algorithm for generating initial code guesses. We focus on the case of a binary tree for simplicity, but more general trees will provide better performance while retaining a significant reduction in complexity.

Say that we have a good rate 1 code as in Fig. 5c and we form a new rate two code by splitting the two codewords as in Fig. 5d. Instead of running a full search VQ design on the resulting 4-word codebook, however, we divide the training sequence into two pieces, collecting together all those vectors encoded into a common word in the 1 bit codebook, that is, all of the training sequence vectors in a common cell of the Voronoi partition. For each of these subsequences of training vectors, we then find a good 1-bit code using the algorithm. The final codebook (so far) consists of the four codewords in the two 1-bit codebooks designed for the two subsequences. A tree-searched encoder selects one of the words not by an ordinary full search of this codebook, but instead it uses the first one bit codebook designed on the whole sequence to select a second code and it then picks the best word in the second code. This encoder can then be used to further subdivide the training sequence and construct even better codebooks for the subsequences. The encoder operation can be depicted as a tree in Fig. 6.

The tree is designed one layer at a time; each new layer being designed so that the new codebook available from each node is good for the vectors encoded into the node. Observe that there are 2^R possible reproduction vectors as in the full search VQ, but now R binary searches are made instead of a single 2^R-ary search. In addition, the encoder

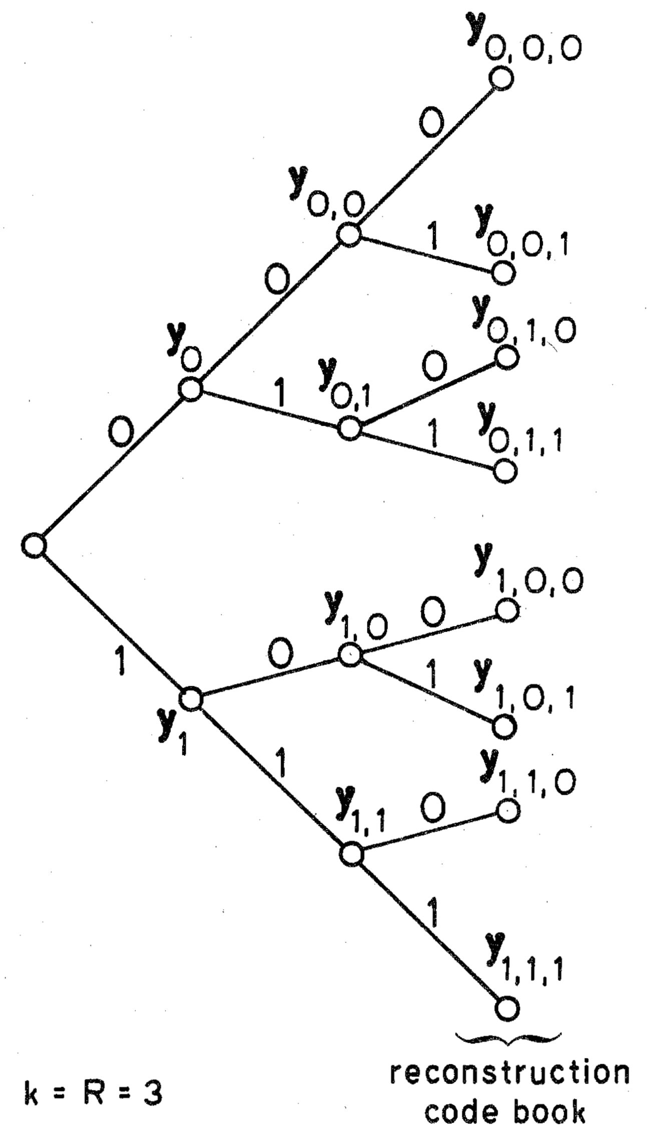


Figure 6. Tree-Searched VQ. A binary encoder tree is shown for a three-dimensional one bit per sample VQ. The encoder makes a succession of R minimum distortion choices from binary codebooks, where the available codebook at each level consists of labels of the nodes in the next level. The labels of the nodes of the final layer are the actual reproduction codewords. At each node the encoder chooses the minimum distortion available label and, if the new index is a 0 (1), sends a channel symbol of 0 (1) and advances up (down) to the next node. After R binary selections the complete channel codeword has been sent and the reproduction codeword specified to the decoder.

storage requirements have doubled. The encoder is no longer optimal for the decoder in the sense of Property 1 since it no longer can perform an exhaustive search of the codebook. The search, however, is much more efficient if done sequentially than is a full search. Thus one may trade performance for efficiency of implementation.

Nonbinary trees can also be used where at the i^{th} layer codebooks of rate R_i are used and the overall rate is then $\Sigma_i R_i$. For example, a depth three tree for VQ of LPC parameter vectors using successive rates of 4, 4, and 2 bits per vector yields performance nearly as good as a full search VQ of the same total rate of 10 bits per vector, yet for the tree search one need only compute $2^4 + 2^4 + 2^2 = 36$ distortions instead of $2^{10} = 1028$ distortions [10].

Other techniques can be used to design tree-searched codes. For example, Adoul *et al.* [32] use a separating hyperplane approach. Another approach is to begin with a full search codebook and to design a tree-search into the codebook. One technique for accomplishing this is to first group the codewords into close disjoint pairs and then form the centroids of the pairs as the node label of the immediate ancestor of the pair. One then works backwards through the tree, always grouping close pairs. Ideally, one would like a general design technique for obtaining a tree search into an arbitrary VQ codebook with only a small loss of average distortion. Gersho and Cheng [33] have reported preliminary results for designing a variable-length tree search for an arbitrary codebook and have demonstrated its implementability for several small dimensional examples.

Multistep VQ

A multistep VQ is a tree-searched VQ where only a single small codebook is stored for each layer of the tree instead of a different codebook for each node of each layer. Such codes provide the computation reduction of tree-searched codes while reducing the storage requirements below that of even ordinary VQ's. The first example of such a code was the multistage codebook [34]. For simplicity we again confine interest to codes which make a sequence of binary decisions. The first layer binary code is designed as in the tree-searched case. This codebook is used to encode the training sequence and then a training sequence of error or residual vectors is formed. For waveform coding applications the error vectors are simply the difference of the input vectors and their codewords. For vocoding applications, the error vectors are residuals formed by passing the input waveform through the inverse filter $A(z)/\alpha$. The algorithm is then run to design a binary VQ for this vector training sequence of coding errors. The reconstruction for these two bits is then formed by combining the two codewords: For waveform coding this is accomplished by adding the first codeword to the error codeword. For voice coding this is accomplished by using the cascade of two all-pole filters for synthesis. This reproduction can then be used to form a "finer" error vector and a code designed for it. Thus an input vector is encoded in stages as with the tree-searched code, but now only R binary codebooks and hence $2R$ total codewords need to be stored. Observe that there are still 2^R possible final codewords, but we have not needed this much storage because the code can be constructed by adding different combinations of a smaller set of words. A multistage VQ is depicted in Fig. 7.

Product codes

Another useful structure for a memoryless VQ is a prod-

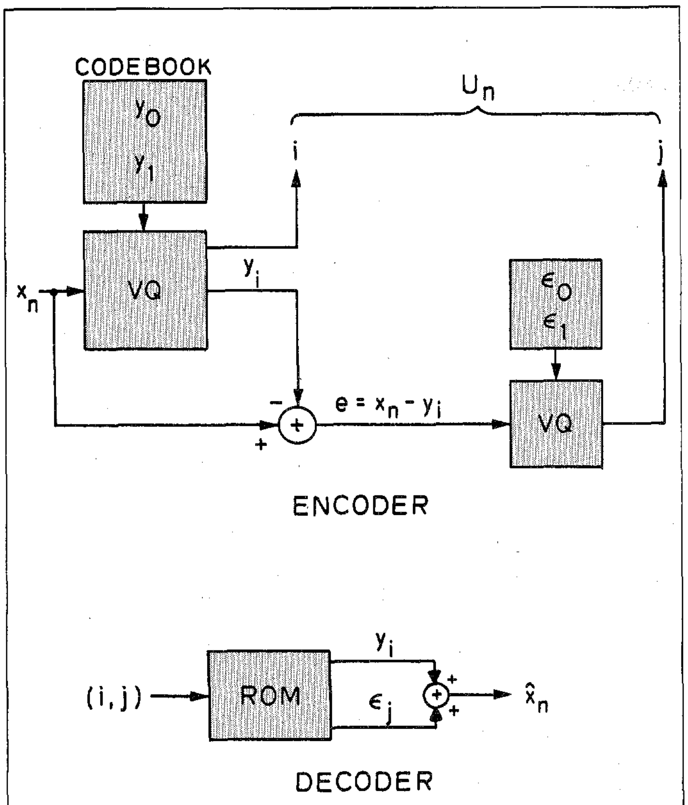


Figure 7. Multistage VQ with 2 Stages. The input vector is first encoded by one VQ and an error vector is formed. The second VQ then encodes the error vector. The two channel symbols from the two VQ's together form the complete channel symbol for the entire encoder. The decoder adds together the corresponding reproduction vectors.

uct code. In one extreme, multiple use of scalar quantizers is equivalent to product VQ's and are obviously simple to implement. More general product VQ's, however, may permit one to take advantage of the performance achievable by VQ's while still being able to achieve the higher rates required for good fidelity. In addition, such codes may yield a smaller computational complexity than an ordinary VQ of the same rate and performance (but different dimension). The basic technique is useful when there are differing aspects of the input vector that one might wish to code separately because of different effects, e.g., on dynamic range or finite word length implementation.

Gain/shape VQ

One example of a product code is a gain/shape VQ where separate, but interdependent, codes are used to code the "shape" and "gain" of the waveform, where the "shape" is defined as the original input vector normalized by removal of a "gain" term such as energy in a waveform coder or LPC residual energy in a vocoder. Gain/shape encoders were introduced by Buzo *et al.* [16] and were subsequently extended and optimized by Sabin and Gray [35,36]. A gain/shape VQ for waveform coding with a squared-error distortion is illustrated in Fig. 8.

Figure 8 sketches the surprising fact that for the squared error case considered, the two-step selection of the product codeword is an optimal encoding for the given product codebook. We emphasize that here the encoder is optimal for the given product codebook or decoder, but the codebook itself is in general suboptimal because of the constrained product form. A similar property holds for the Itakura-Saito distortion gain/shape VQ. Thus in this case if one devotes R_s bits to the shape and R_g bits to the gain, where $R_s + R_g = R$, then one need only compute 2^{R_s} vector distortions and an easy scalar quantization. The full search encoder would require 2^R vector distortions, yet both encoders yield the same minimum distortion codeword!

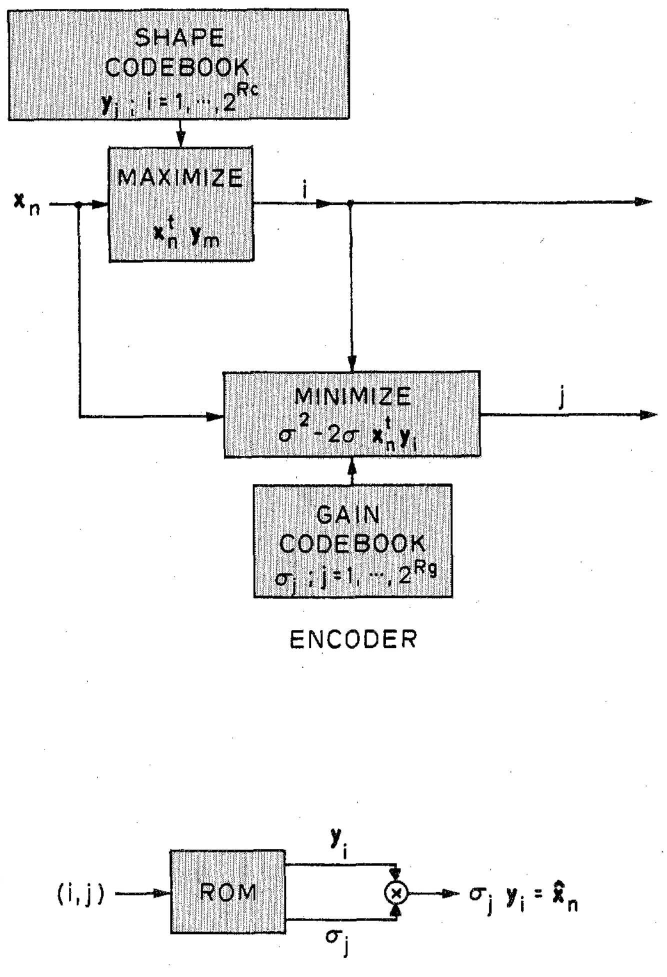

SHAPE CODEBOOK
$y_i\ ;\ i=1,\cdots,2^{R_c}$
MAXIMIZE
$x_n^t\ y_m$
x_n
i
MINIMIZE
$\sigma^2 - 2\sigma\ x_n^t y_i$
j
GAIN CODEBOOK
$\sigma_j\ ;\ j=1,\cdots,2^{R_g}$
ENCODER
y_i
(i,j)
ROM
σ_j
$\sigma_j\ y_i = \hat{x}_n$
DECODER

Figure 8. Gain/Shape VQ. First a unit energy shape vector is chosen to match the input vector by maximizing the inner product over the codewords. Given the resulting shape vector, a scalar gain codeword is selected so as to minimize the indicated quantity. The encoder yields the product codeword $\sigma_j y_i$ with the minimum possible squared error distortion from the input vector. Thus this multistep encoder is optimum for the product codebook.

Variations of the basic VQ algorithm can be used to iteratively improve a gain shape code by alternately optimizing the shape for the gain and vice versa. The resulting conditional centroids are easy to compute. The centroid updates can be made either simultaneously or alternately after each iteration [36].

One can experimentally determine the optimal bit allocation between the gain and the shape codebooks.

Separating mean VQ

Another example of a multistep product code is the separating mean VQ where a sample mean instead of an energy term is removed [37]. Define the sample mean $\langle \mathbf{x} \rangle$ of a k-dimensional vector by $k^{-1} \sum_{i=0}^{k-1} x_i$. In a separated mean VQ one first uses a scalar quantizer to code the sample mean of a vector, then the coded sample mean is subtracted from all of the components of the input vector to form a new vector with approximately zero sample mean. This new vector is then vector quantized. Such a system is depicted in Fig. 9. The basic motivation here is that in image coding the sample mean of pixel intensities in a small rectangular block represents a relatively slowly varying average background value of pixel intensity around which there are variations.

To design such a VQ, first use the algorithm to design a scalar quantizer for the sample mean sequence $\langle \mathbf{x}_j \rangle$, $j = 0,1,\ldots,L-1$. Let $q(\langle \mathbf{x} \rangle)$ denote the reproduction for $\langle \mathbf{x} \rangle$ using the quantizer. Then use the vector training sequence $\mathbf{x}_j - q(\langle \mathbf{x}_j \rangle)\mathbf{1}$, where $\mathbf{1} = (1,1,\ldots,1)$, to design a VQ for the difference. Like the gain/shape VQ, a product codebook and a multistep encoder are used, but unlike the gain/shape VQ it can be shown that the multistep encoder here does not select the best possible mean, shape pair, that is, the multistep encoder is not equivalent to a full search encoder.

Lattice VQ

A final VQ structure capable of efficient searches and memory usage is the lattice quantizer, a k-dimensional generalization of the scalar uniform quantizer. A lattice in k-dimensional space is a collection of all vectors of the form $y = \sum_{i=0}^{n-1} a_i \mathbf{e}_i$, where $n \leq k$, where $\mathbf{e}_0, \ldots, \mathbf{e}_{n-1}$ are a set of linearly independent vectors in $\mathbf{R}^k$, and where the a_i are arbitrary integers. A lattice quantizer is a quantizer whose codewords form a subset of a lattice. Lattice quantizers were introduced by Gersho [38] and the performance and efficient coding algorithms were developed for many particular lattices by Conway and Sloane [39,40,41] and Barnes and Sloane [42]. The disadvantage of lattice quantizers is that they cannot be improved by a variation of the Lloyd algorithm without losing their structure and good quantizers produced by the Lloyd algorithm cannot generally be well approximated by lattices. Lattice codes can work well on source distributions that are approximately uniform over a bounded region of space. In fact, lattices that are asymptotically optimal in the limit of large rate are known for this case in two and three dimensions and good lattices are known for dimensions up to 16.

Ideally, one would like to take a full search, unconstrained VQ and find some fast means of encoding having complexity more like the above techniques than that of the full search. For example, some form of multidimensional companding followed by a lattice quantizer as suggested by Gersho [24] would provide both good performance and efficient implementation. Unfortunately, however, no design methods accomplishing this goal have yet been found.

FEEDBACK VECTOR QUANTIZERS

Memory can be incorporated into a vector quantizer in a simple manner by using different codebooks for each input vector, where the codebooks are chosen based on past input vectors. The decoder must know which codebook is being used by the encoder in order to decode the channel symbols. This can be accomplished in two ways: 1) The encoder can use a codebook selection procedure that depends only on past encoder outputs and hence the codebook sequence can be tracked by the decoder. 2) The decoder is informed of the selected codebook via a special low-rate side channel. The first approach is called feedback vector quantization and is the

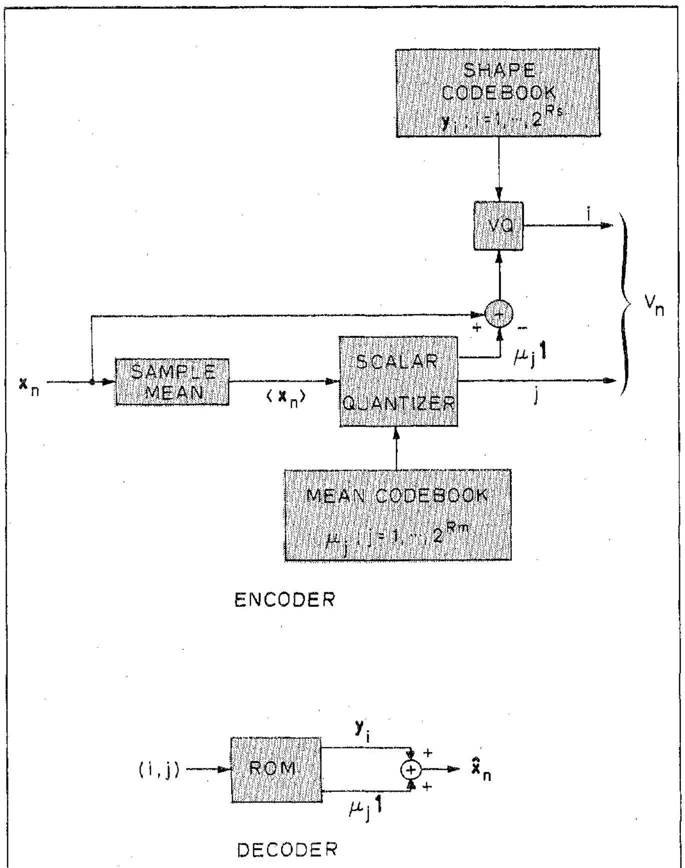


Figure 9. Separating Mean VQ. The sample mean of the input vector is computed, scalar quantized, and then subtracted from each component of the input vector. The resulting vector with approximately zero sample mean is then vector quantized. The decoder adds the coded sample mean to all components of the coded shape vector.

topic of this section. The name follows because the encoder output is "fed back" for use in selecting the new codebook. A feedback vector quantizer can be viewed as the vector extension of a scalar adaptive quantizer with backward estimation (AQB) [3]. The second approach is the vector extension of a scalar adaptive quantizer with forward estimation (AQF) and is called simply adaptive vector quantization. Adaptive VQ will be considered in a later section. Observe that systems can combine the two techniques and use both feedback and side information. We also point out that unlike most scalar AQB and AQF systems, the vector analogs considered here involve no explicit estimation of the underlying densities.

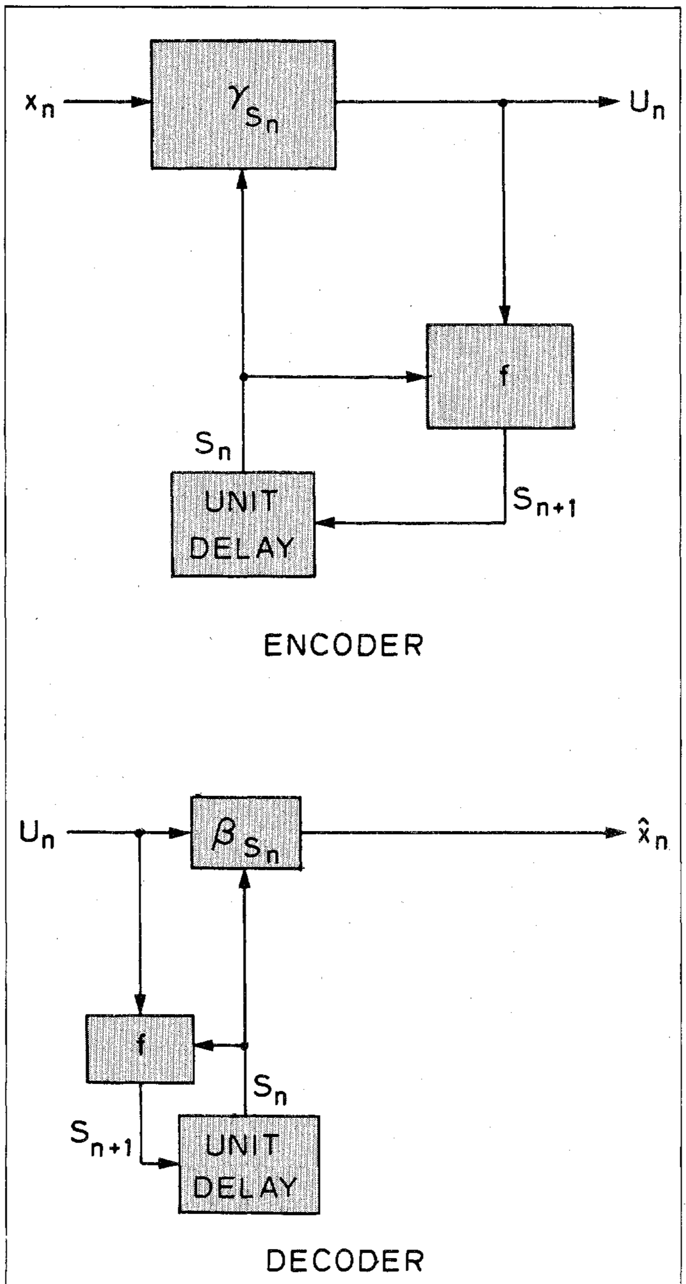


Figure 10. Feedback VQ. At time n both encoder and decoder are in a common state S_n. The encoder uses a state VQ γ_{S_n} to encode the input vector and then selects a new state for the next input vector. Knowing the VQ used and the resulting channel symbol, the decoder can produce the correct reproduction. Note that the state VQ's may be computed at each time from some rule or, if they are small in number, simply stored separately.

It should be emphasized that the results of information theory imply that VQ's with memory can do no better than memoryless VQ's in the sense of minimizing average distortion for a given rate constraint. In fact, the basic mathematical model for a data compression system in information theory is exactly a memoryless VQ and such codes can perform arbitrarily close to the optimal performance achievable using any data compression system. The exponential growth of computation and memory with rate, however, may result in nonimplementable VQ's. A VQ with memory may yield the desired distortion with practicable complexity.

A general feedback VQ can be described as follows [22]: Suppose now that we have a space **S** whose members we shall call states and that for each state s in **S** we have a separate quantizer: an encoder γ_s, decoder β_s, and codebook C_s. The channel codeword space **M** is assumed to be the same for all of the VQ's. Consider a data compression system consisting of a sequential machine such that if the machine is in state s, then it uses the quantizer with encoder γ_s and decoder β_s. It then selects its next state by a mapping called a next-state function or state-transition function f such that given a state s and a channel symbol v, then $f(v,s)$ is the new state of the machine. More precisely, given a sequence of input vectors $\{\mathbf{x}_n;\ n = 0,1,2,\ldots\}$ and an initial state s_0, then the subsequent state sequence s_n, channel symbol sequence v_n, and reproduction sequence $\hat{\mathbf{x}}_n$ are defined recursively for $n = 0,1,2,\ldots$ as

$$v_n = \gamma_{s_n}(\mathbf{x}_n), \qquad \hat{\mathbf{x}}_n = \beta_{s_n}(v_n), \qquad s_{n+1} = f(v_n, s_n). \quad (5)$$

Since the next state depends only on the current state and the channel codeword, the decoder can track the state if it knows the initial state and the channel sequence. A general feedback vector quantizer is depicted in Fig. 10. The freedom to use different quantizers based on the past without increasing the rate should permit the code to perform better than a memoryless quantizer of the same dimension and rate.

An important drawback of all feedback quantizers is that channel errors can accumulate and cause disastrous reconstruction errors. As with scalar feedback quantizer systems, this must be handled by periodic resetting or by error control or by a combination of the two.

If the state space is finite, then we shall call the resulting system a finite-state vector quantizer or FSVQ. For an FSVQ, all of the codebooks and the next-state transition table can all be stored in ROM, making the general FSVQ structure amenable to LSI or VLSI implementation [43].

Observe that a memoryless vector quantizer is simply a feedback vector quantizer or finite-state vector quantizer with only a single state. The general FSVQ is a special case of a tracking finite state source coding system [44] where the encoder is a minimum distortion mapping.

Three design algorithms for feedback vector quantizers using variations on the generalized Lloyd algorithm have been recently developed. The remainder of this section is devoted to brief descriptions of these techniques.

Vector predictive quantization

Cuperman and Gersho [45,46] proposed a vector predictive coder or vector predictive quantizer (VPQ) which is a vector generalization of DPCM or predictive quantization. A VPQ is sketched in Fig. 11. For a fixed predictor, the VQ design algorithm is used to design a VQ for the prediction error sequence. Cuperman and Gersho considered several variations on the basic algorithm, some of which will be later mentioned.

Chang [47] developed an extension to Cuperman and Gersho's algorithm which begins with their system and then uses a stochastic gradient algorithm to iteratively improve the vector linear predictor coefficients, that is, to better match the predictor to the quantizer. The stochastic gradient algorithm is used only in the design of the system, not as an on line adaptation mechanism as in the adaptive gradient algorithms of, e.g., Gibson *et al.* [48] and Dunn [49]. A scalar version of this algorithm for improving the predictor for the quantizer was developed in unpublished work of Y. Linde.

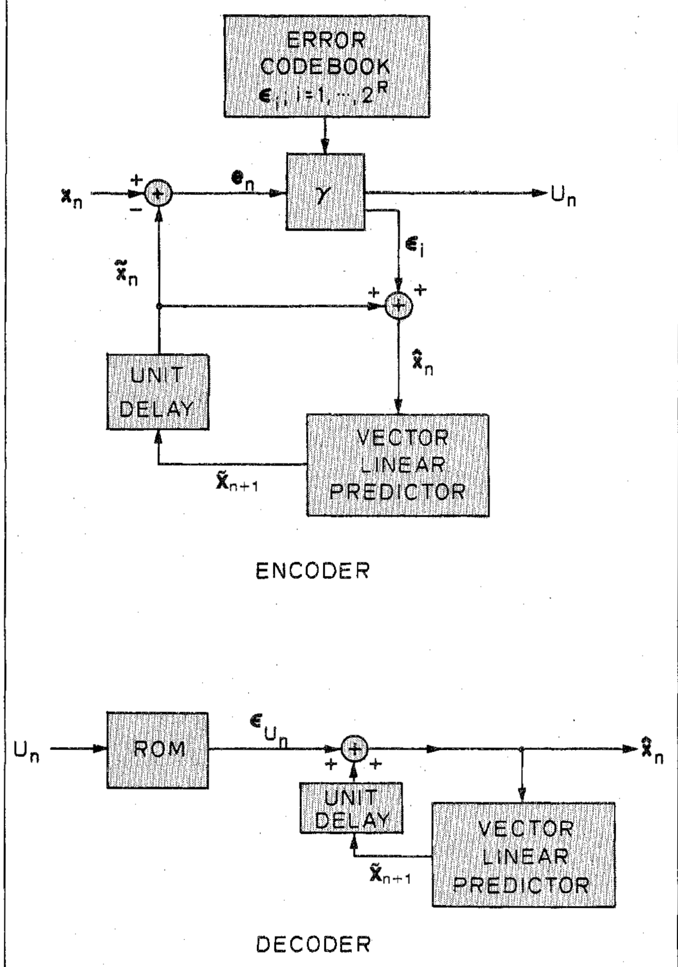


Figure 11. Vector Predictive Quantization. A linear vector predictor for the next input vector of a process given the previous input vector is applied to the previous reproduction of the input vector. The resulting prediction is subtracted from the current input vector to form an error vector which is vector quantized. The decoder uses a copy of the encoder and the received encoded error vectors to construct the reproduction.

Product/multistep FVQ

A second basic approach for designing feedback vector quantizers which is quite simple and works quite well is to use a product multistep VQ such as the gain/shape VQ or the separating mean VQ and use a simple feedback quantizer on the scalar portion and an ordinary memoryless VQ on the remaining vector. This approach was developed in [10] for gain/shape VQ of LPC parameters and in [37] for separating mean VQ of images. Both efforts used simple scalar predictive quantization for the feedback quantization of the scalar terms.

FSVQ

The first general design technique for finite-state vector quantizers was reported by Foster and Gray [50,51]. There are two principal design components: 1. Design an initial set of state codebooks and a next-state function using an *ad hoc* algorithm. 2. Given the next-state function, use a variation of the basic algorithm to attempt to improve the state codebooks. The second component is accomplished by a slight extension of the basic algorithm that is similar to the extension of [52] for the design of trellis encoders: Encode the data using the FSVQ and then replace all of the reproduction vectors by the centroids of the training vectors which map into those vectors; now, however, the centroids are conditioned on both the channel symbol and the state. While such conditional averages are likely impossible to compute analytically, they are easily computed for a training sequence. For example, in the case of a squared error distance one simply forms the Euclidean centroid of all input vectors which correspond to the state s and channel symbol v in an encoding of the training sequence.

As with ordinary VQ, replacing the old decoder or codebook by centroids cannot yield a code with larger distortion. Unlike memoryless VQ, however, replacing the old encoder by a minimum distortion rule for the new decoder can in principal cause an increase in distortion and hence now the iteration is somewhat different: Replace the old encoder (which is a minimum distortion rule for the old decoder) by a minimum distortion rule for the new decoder. If the distortion goes down, then continue the iteration and find the new centroids. If the distortion goes up, then quit with the encoder being a quantizer for the previous codebook and the decoder being the centroids for the encoder. By construction this algorithm can only improve performance. It turns out, however, that in practice it is a good idea to not stop the algorithm if the distortion increases slightly, but to let it continue: it will almost always eventually drop back down in distortion and converge to something better.

The first design component is more complicated. We here describe one of the more promising approaches of [51] called the omniscient design approach. Say that we wish to design an FSVQ with K states and rate R bits per vector. For simplicity we label the states as 0 through K-1. First use the training sequence to design a memoryless VQ with K codewords, one for each state. We shall call these codewords state labels and this VQ the state quantizer. We call the output of the state VQ the "ideal next state" instead of a channel symbol. Next break up the training sequence into subsequences as follows: Encode the training sequence using the state VQ and for each state s collect all of training vectors which *follow* the occurrence of this state label. Thus for s the corresponding training subsequence consists of all input vectors that occur when the *current* ideal state is s. Use the basic algorithm to design a rate R codebook C_s for the corresponding training sequence for each s.

The resulting state VQ and the collection of codebooks for each state have been designed to yield good performance in the following communication system: The encoder is in an ideal state s chosen by using the state VQ on the last input vector. The encoder uses the corresponding VQ encoder γ_s described by the codebook C_s. The output of γ_s is the channel symbol. In order to decode the channel symbol, the decoder must also know the ideal state. Unfortunately, however, this ideal state cannot be determined from knowledge of the initial state and all of the received channel symbols. Thus the decoder must be omniscient in the sense of knowing this additional side information in order to be able to decode. In particular, this system is not an FSVQ by our definition. We can use the state quantizer and the various codebooks, however, to construct an FSVQ by approximating the omniscient system: Instead of forming the ideal next state by using the state VQ on the actual input vector (as we did in the design procedure), use the state VQ on the current reproduction vector in order to choose the next state. This will yield a state sequence depending only on encoder outputs and the original state and hence will be trackable by the decoder. This is analogous to the scalar practice of building a predictive coder and choosing the predictor as if it knew the past inputs, but in fact applying it to past reproductions.

Combining the previously described steps of (1) initial (state label) codebook design, (2) state codebooks and next-state function design, and (3) iterative improvement of code for given next-state function, provides a complete design algorithm.

In addition to the above design approach, techniques have been developed for iterating on (2) and (3) above in the sense of optimizing the next-state function for a given collection of codebooks. These algorithms, however, are more complicated and require ideas from the theory of adaptive stochastic automata. The reader is referred to [53] for a discussion of these improvement algorithms.

VECTOR TREE AND TRELLIS ENCODERS

As with scalar feedback quantizers, the actions of the decoder of a feedback VQ can be depicted as a directed graph or tree. A simple example is depicted in Fig. 12, where a merged tree or trellis can be drawn since the feedback VQ has only a finite number of states.

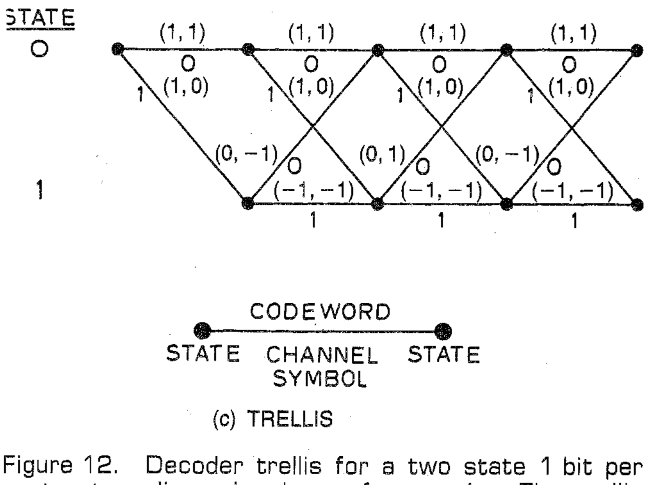
Instead of using the ordinary VQ encoder which is only permitted to look at the current input vector in order to decide on a channel symbol, one could use algorithms such as the Viterbi algorithm, M-algorithm or M,L-algorithm, Fano algorithm, or stack algorithm for a minimum cost search through a directed graph and search several levels ahead into the tree or trellis before choosing a channel symbol. This introduces an additional delay into the encoding of several vectors, but it ensures better long run average distortion behavior. This technique is called tree or trellis encoding and is also referred to as look-ahead coding, delayed decision coding, and multipath search coding. (See, e.g., [54, 52] for surveys.) We point out that a tree encoding system uses a tree to denote the

s	u	β_s (u)
0	0	(1,1)
0	1	(1,0)
1	0	(0,-1)
1	1	(-1,-1)

(a) DECODER

s	u	f (u,s)
0	0	0
0	1	1
1	0	0
1	1	1

(b) NEXT-STATE FUNCTION



(c) TRELLIS

Figure 12. Decoder trellis for a two state 1 bit per vector two dimensional waveform coder. The trellis depicts the possible state transitions for the given next-state function. The transitions are labeled by the corresponding decoder output (in parentheses) and channel symbol produced by the encoder.

operation on successive vectors by the decoder at successive times while a tree-searched VQ uses a tree to construct a fast search for a single vector at a single time.

A natural variation of the basic algorithm for designing FSVQ's can be used to design trellis encoding systems: Simply replace the FSVQ encoder which finds the minimum distortion reproduction for a single input vector by a Viterbi or other search algorithm which searches the decoder trellis to some fixed depth to find a good long term minimum distortion path. The centroid computation is accomplished exactly as with an FSVQ: each branch or transition label is replaced by the centroid of all training vectors causing that transition, that is, the centroid conditioned on the decoder state and channel symbol. Scalar and simple two dimensional vector trellis encoding systems were designed in [52] using this approach.

Trellis encoding systems are not really vector quantization systems as we have defined them since the encoder is permitted to search ahead to determine the effect on the decoder output of several input vectors while a vector quantizer is restricted to search only a single vector ahead. The two systems are intimately related, however, and a trellis encoder can always be used to improve the performance of a feedback vector quantizer. Very little work has yet been done on vector trellis encoding systems.

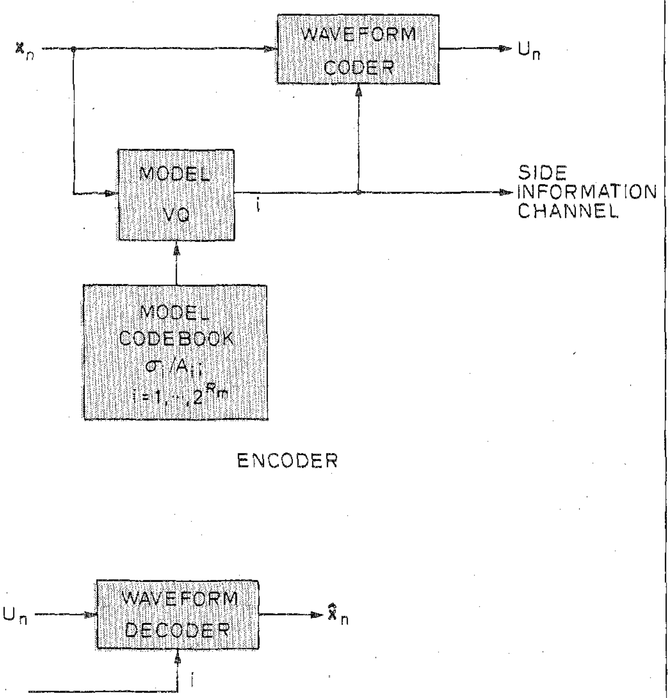


Figure 13. Adaptive VQ. The model VQ uses the Itakura-Saito distortion to select an LPC model to fit the input frame of many sample vectors. This selection in turn determines the waveform coder used to digitize the sample vectors. A side channel then informs the receiver which decoder to use on the channel symbols produced by the waveform coder.

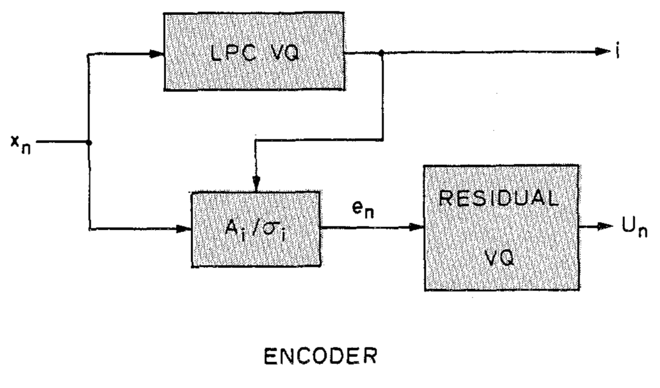
ADAPTIVE VQ

As a final class of VQ we consider systems that use one VQ to adapt a waveform coder, which might be another VQ. The adaptation information is communicated to the receiver via a low rate side information channel.

The various forms of vector quantization using the Itakura-Saito family of distortion measures can be considered as model classifiers, that is, they fit an all-pole model to an observed sequence of sampled speech. When used alone in an LPC VQ system, the model is used to synthesize the speech at the receiver. Alternatively, one could use the model selected to choose a waveform coder designed to be good for sampled waveforms that produce that model. For example, analogous to the omniscient design of FSVQ one could design separate VQ's for the subsequences of the training sequence encoding into common models. Both the model index and the waveform coding index are then sent to the receiver. Thus LPC VQ can be used to adapt a waveform coder, possibly also a VQ or related system. This will yield a system typically of much higher rate, but potentially of much better quality since the codebooks can be matched to local behavior of the data. The general structure is shown in Fig. 13. The model VQ typically operates on a much larger vector of samples and at a much lower rate in bits per sample than does the waveform coder and hence the bits spent on specifying the model through the side channel are typically much fewer than those devoted to the waveform coder.

There are a variety of such possible systems since both the model quantizer and the waveform quantizer can take on many of the structures so far considered. In addition, as in speech recognition applications [55] the gain-independent variations of the Itakura-Saito distortion measure which either normalize or optimize gain may be better suited for the model quantization than the usual form. Few such systems have yet been studied in detail. We here briefly describe some systems of this type that have appeared in the literature to exemplify some typical combinations. All of them use some form of memoryless VQ for the model quantization, but a variety of waveform coders are used.

The first application of VQ to adaptive coding was by Adoul, Debray, and Dalle [32] who used an LPC VQ to choose a predictor for use in a scalar predictive waveform coder. Vector quantization was used only for the adaptation and not for the waveform coding. An adaptive VQ generalization of this system was later developed by Cuperman and Gersho [45,46] who used an alternative classification technique to pick one of three vector predictors and then used those predictors in a predictive vector quantizer. The predictive vector quantizer design algorithm previously described was used, except now the training sequence was broken up into subsequences corresponding to the selected predictor and a quantizer was designed for each resulting error sequence. Chang [47] used a similar scheme with an ordinary LPC VQ as the classifier and with a stochastic gradient algorithm run on each of the vector predictive quantizers in order to im-



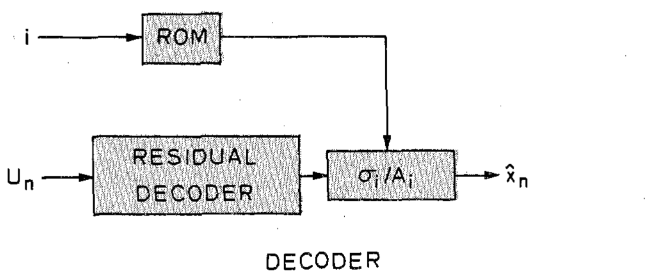


Figure 14. RELP VQ. An LPC VQ is used for model selection and a single VQ to waveform encode the residuals formed by passing the original waveform through the inverse filter $A/\sqrt{\alpha}$. The side information specifies to the decoder which of the model filters $\sqrt{\alpha}/A$ should be used for synthesis.

prove the prediction coefficients for the corresponding codebooks.

Rebolledo *et al.* [56] and Adoul and Mabilleau [57] developed vector residual excited linear predictive (RELP) systems. (See Fig. 14.) A similar system employing either a scalar or a simple vector trellis encoder for the waveform coder was developed by Stewart *et al.* [52]. Both of these systems used the basic algorithm to design both the model VQ and the waveform coders.

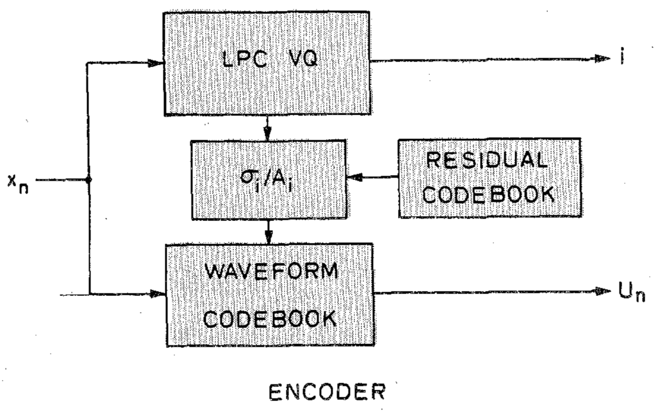
The RELP VQ systems yielded disappointingly poor performance at low bit rates. Significantly better performance was achieved by using the residual codebooks produced in the RELP design to construct codebooks for the original waveform, that is, instead of coding the model and the residual, code the model and use the selected model to construct a waveform coder for the original waveform as depicted in Fig. 15 [52]. For lack of a better name, this system might be called an inverted RELP because it uses residual codebooks to drive an inverse model filter in order to get a codebook for the original waveform.

Yet another use of LPC VQ to adapt a waveform coder was reported by Heron, Crochiere, and Cox [58] who used a subband/transform coder for the waveform coding and used the side information to adapt the bit allocation for the scalar parameter quantizers.

Many other variations on the general theme are possible and the structure is a promising one for processes such as speech that exhibit local stationarity, that is, slowly varying short term statistical behavior. The use of one VQ to partition a training sequence in order to design good codes for the resulting distinct subsequences is an intuitive approach to the computer-aided design of adaptive data compression systems.

EXAMPLES

We next consider the performance of various forms of vector quantizers on three popular guinea pigs: Gauss Markov sources, speech waveforms, and images. For the speech coding example we consider both waveform coders using the squared error distortion measure and vocoders using the Itakura-Saito distortion. The caveats of the introduction should be kept in mind when interpreting the results.



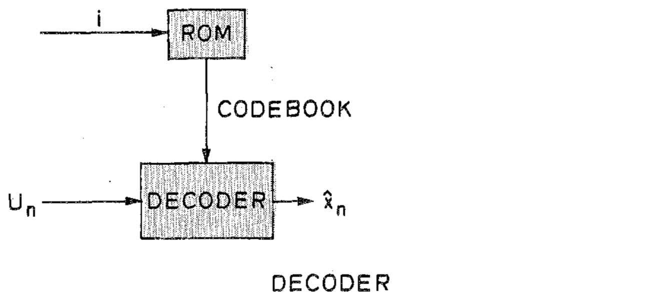


Figure 15. Inverted RELP. An LPC VQ is used to select a model filter σ/A. A waveform codebook is then formed by driving the model filter with all possible residual codewords from a RELP VQ design. Thus, unlike a RELP system, the original waveform (and not a residual) is matched by possible reproduction codewords.

The performance of the systems are given by SNR's for squared error and by an analogous quantity for the Itakura-Saito distortion: In both cases we measure normalized average distortion on a logarithmic scale, where the normalization is by the average distortion of the optimum zero rate code—the average distortion between the input sequence and the centroid of the entire input sequence. This quantity reduces to an SNR in the squared error case and provides a useful dimensionless normalized average distortion in general. We call this quantity the SNR in both cases. The SNR is given in tables instead of graphs in order to facilitate quantitative comparisons among the coding schemes.

Gauss Markov sources

We first consider the popular guinea pig of a Gauss Markov source. This source is useful as a mathematical model for some real data sources and its information theoretic optimal performance bounds as described by the distortion-rate function are known. For this example we consider only the squared error distortion. A Gauss Markov source or a first order Gauss autoregressive source $\{X_n\}$ is defined by the difference equation $X_{n+1} = aX_n + W_n$, where $\{W_n\}$ is a zero mean, unit variance, independent and identically distributed Gaussian source. We here consider the highly correlated case of $a = 0.9$ and vector quantizers of 1 bit/sample. The maximum achievable SNR as given by Shannon's distortion-rate function for this source and rate is 13.2 dB [7].

Various design algorithms were used to design vector quantizers for several dimensions for this source. Table I describes the results of designing several memoryless vector quantizers for a training sequence of 60,000 samples. Given are the design SNR (code performance on the training sequence), the number of multiplications per sample required by the encoder, and the number of real scalars that must be stored for the encoder codebook. The number of multiplications is used as a measure of encoder complexity because it is usually the dominant computation and because the number of additions required is usually comparable. It is given by $n =$ (the number of codewords searched) $\times$ (dimension)/(dimension) = the number of codewords searched. The actual storage required depends on the number of bytes used to store each floating point number. Many (but not all) of the final codes were subsequently tested on different test sequences of 60,000 samples. In all cases the open test SNR's were within .25 dB of the design distortion. The systems considered are full search VQ's [25], binary tree-searched VQ's [59], binary multistage VQ's [47], and gain/shape VQ's [36]. The gain and codebook sizes for the gain/shape codes were experimentally optimized.

As expected, the full search VQ yields the best performance for each dimension, but the tree-searched VQ is not much worse and has a much lower complexity. The multistage VQ is noticeably inferior, losing more than 1 dB at the higher dimensions, but its memory requirements are small. The gain/shape VQ compares poorly on the basis of performance vs. rate for a fixed dimension, but it is the best code in the sense of providing the minimum distortion for a fixed complexity and rate.

For larger rates and lower distortion the relative merits may be quite different. For example, the multistage VQ is then capable of better performance relative to the ordinary VQ since the quantization errors in the various stages do not accumulate so rapidly. (See, e.g., [34].) Thus in this

TABLE I

MEMORYLESS VQ FOR A GAUSS MARKOV SOURCE.

	VQ			TSVQ			MVQ			G/SVQ		
k	SNR	n	M	SNR	n	M	SNR	n	M	SNR	n	M
1	4.4	2	2	4.4	2	2	4.4	2	2			
2	7.9	4	8	7.9	4	12	7.6	4	8	7.9	1	3
3	9.2	8	24	9.2	6	42	8.6	6	18	9.3	1	5
4	10.2	16	64	10.2	8	120	8.4	8	32	9.4	2	10
5	10.6	32	160	10.4	10	310	9.3	10	50	9.8	3	17
6	10.9	64	384	10.7	12	756	9.1	12	72	9.9	4	26
7	11.2	128	896	11.0	14	1778	9.4	14	98	10.2	4	31
8							9.9	16	128	10.6	5	43
9										10.9	6	57

Signal to Noise Ratios (SNR), number of multiplications per sample (n), and storage requirements of memoryless vector quantizers: full search memoryless VQ (VQ), binary tree-searched (TSVQ), binary multistage VQ (MVQ), and gain/shape VQ (G/SVQ). Rate = 1 bit/sample. k = vector dimension. Training Sequence = 60000 samples from a Gauss Markov Source with correlation coefficient 0.9.

TABLE II

FEEDBACK VQ OF A GAUSS MARKOV SOURCE.

	FSVQ1				FSVQ2				VPQ		
k	SNR	K	n	M	SNR	K	n	M	SNR	n	M
1	10.0	64	2	64	9.5	16	2	16	10.0	2	2
2	10.8	256	4	512	10.8	32	4	64	11.2	4	8
3	11.4	512	8	1536	11.1	64	8	192	11.6	8	24
4	12.1	512	16	2048	11.3	128	16	512	11.6	16	64

Signal to Noise Ratios (SNR), number of states (K), number of multiplications per sample (n), and storage (M) for feedback quantizers: FSVQ with number of states increased until negligible change (FSVQ1), FSVQ with fewer states (FSVQ2), VPQ. Rate = 1 bit/sample. k = vector dimension. Training Sequence = 60000 samples from a Gauss Markov Source with correlation coefficient 0.9.

TABLE III

MEMORYLESS VQ OF SAMPLED SPEECH.

	VQ				TSVQ			
k	SNRin	SNRout	n	M	SNRin	SNRout	n	M
1	2.0	2.1	2	2	2.0	2.1	2	2
2	5.2	5.3	4	8	5.1	5.1	4	12
3	6.1	6.0	8	24	5.5	5.5	6	42
4	7.1	7.0	16	64	6.4	6.4	8	120
5	7.9	7.6	32	160	7.1	6.9	10	310
6	8.5	8.1	64	384	7.9	7.5	12	756
7	9.1	8.4	128	896	8.3	7.8	14	1778
8	9.7	8.8	256	2048	8.9	8.0	16	4080

	MVQ				G/SVQ			
k	SNRin	SNRout	n	M	SNRin	SNRout	n	M
1	2.0	2.1	2	2				
2	4.3	4.4	4	8				
3	4.3	4.4	6	18	4.5	4.6	4	14
4	4.4	4.5	8	32	6.0	6.1	4	20
5	5.0	5.0	10	50	7.2	6.9	8	44
6	5.0	4.9	12	72	7.7	7.4	16	100
7	5.3	5.1	14	98	8.2	7.7	16	120
8	5.6	5.5	16	128	8.8	8.1	32	264
9					9.3	8.5	64	584
10					9.8	8.9	128	1288
11					10.4	9.3	256	2824

Signal to Noise Ratios inside training sequence (SNRin) of 640000 speech samples, Signal to Noise Ratios outside training sequence (SNRout) of 76800 speech samples, number of multiplications per sample (n), and storage requirements of memoryless vector quantizers: full search memoryless VQ (VQ), binary tree-searched (TSVQ), binary multistage VQ (MVQ), and gain/shape VQ (G/SVQ). Rate = 1 bit/sample. *k* = vector dimension.

case multistage VQ may be far better because if its much smaller computational requirements.

Table II presents results for three feedback VQ's for the same source. In addition to the parameters of Table I, the number of states for the FSVQ's are given. The first FSVQ and the VPQ were designed for the same training sequence of 60,000 samples. Because of the extensive computation required and the shortness of the training sequence for a feedback quantizer, only dimensions 1 through 4 were considered. The first FSVQ was designed using the omniscient design approach for 1 bit per sample, dimensions 1 through 4, and a variety of numbers of states. For the first example, the number of states was chosen by designing FSVQ's for more and more states until firther increases yielded negligible improvements [51]. It was found, however, that the performance outside of the training sequence for these codes was significantly inferior, by 1 to 2 dB for the larger dimensions. From the discussion of average distortion, this suggests that the training sequence was too short. Hence the second FSVQ design (FSVQ2) was run with a larger training sequence of 128,000 samples and fewer states. The test sequence for these codes always yielded performance within .3 dB of the design value. The VPQ test performance with within .1 dB of the design performance. The scalar predictive quantizer performance and the codebook for the prediction error quantizer are the same as the analytically optimized predictive quantization system of Arnstein [60] run on the same data.

Observe that the scalar FSVQ in the first experiment with 64 states yielded performance quite close to that of the scalar VPQ, which does not have a finite number of states. Intuitively the FSVQ is trying to approximate the infinite state machine by using a large number of states. The VPQ, however, is less complex and requires less memory and hence for this application is superior.

For comparison, the best 1 bit/sample scalar trellis encoding system for this source yields 11.25 dB for this source [52]. The trellis encoding system uses a block Viterbi algorithm with a search depth of 1000 samples for the encoder. It is perhaps surprising that in this example the VPQ and the FSVQ with the short delay of only 4 samples can outperform a Viterbi algorithm with a delay of 1000 samples. It points out, however, two advantages of feedback VQ over scalar trellis encoding systems: 1. The decoder is permitted to be a more general form of finite-state machine than the shift-register based nonlinear filter usually used in trellis encoding systems; and 2. the encoder performs a single full search of a small vector codebook instead of a Viterbi algorithm consisting of a tree search of a sequence of scalar codebooks. In other words, single short vector searches may yield better performance than a "look ahead" sequence of searches of scalar codebooks.

Speech waveform coding

The second set of results considers a training sequence of 640,000 samples of ordinary speech from four different male speakers sampled at 6.5 kHz. The reader is reminded that squared error is not generally a subjectively good distortion measure for speech. Better subjective quality may be obtained by using more complicated distortion measures such as the general quadratic distortion measures with input dependent weightings such as the arithmetic segmented distortions. The VQ design techniques extend to such distortion measures, but the centroid computations are more complicated. (See [30] for the theory and [45, 46] for the application of input-weighted quadratic distortion measures.)

Tables III and IV are the counterparts of Tables I and II for this source. Now, however, the SNR's of the codes on test sequences of samples outside of the training sequence (and by a different speaker) are presented for comparison. In addition, some larger dimensions are con-

TABLE IV
FEEDBACK VQ OF SAMPLED SPEECH.

	FSVQ					VPQ			
k	SNRin	SNRout	K	n	M	SNRin	SNRout	n	M
1	2.0	2.0	2	2	2	2.1	2.6	2	2
2	7.8	7.5	32	4	64	6.4	6.2	4	8
3	9.0	8.3	64	10	192	7.3	6.8	8	24
4	10.9	9.4	512	16	2048	8.0	7.6	16	64
5	12.2	10.8	512	32	2560				

Signal to Noise Ratios inside training sequence (SNRin) of 640000 speech samples, Signal to Noise Ratios outside training sequence (SNRout) of 76800 speech samples, number of states (K), number of multiplications per sample (n), and storage (M) for feedback quantizers: Rate = 1 bit/sample. *k* = vector dimension.

sidered because the longer training sequence made them more trustworthy. Again for comparison, the best known (nonadaptive) scalar trellis encoding system for this source yields a performance of 9 dB [52]. Here the trellis encoder uses the *M*-algorithm with a search depth of 31 samples. The general comparisons are similar to those of the previous source, but there are several differences. The tree-searched VQ is now more degraded in comparison to the full search VQ and the multistage VQ is even worse, about 3 dB below the full search at the largest dimension in comparison to about 1 dB for the Gauss Markov case. The complexity and storage requirements are the same except for the shape/gain VQ where different optimum selections of gain and shape codebook size yield different complexity and storage requirements. The VPQ of dimension 4 is inferior to the trellis encoder and the FSVQ of the same dimension. The four dimensional FSVQ, however, still outperforms the scalar trellis encoder.

Observe that an FSVQ of dimension 4 provides better performance inside and outside the training sequence than does a full search memoryless vector quantizer of dimension 8, achieving better performance with 16 4-dimensional distortion evaluations than with 512 8-dimensional distortion computations. The cost, of course, is a large increase in memory. This, however, is a basic point of FSVQ design—to use more memory but less computation.

LPC VQ (vocoding)

Table V presents a comparison of VQ and FSVQ for vector quantization of speech using the Itakura-Saito distortion measure or, equivalently, vector quantization of LPC speech models [16, 14, 53]. The training sequence and test sequence are as above, but now the input dimension is 128 samples and the output vectors are 10th order all-pole models. The training sequence is now effectively shorter since it contains only 5000 input vectors of this dimension. As a result the test results are noticeably different than the design results. Because of the shortness of the training sequence, only FSVQ's of small dimension and few states were considered.

The table summarizes memoryless VQ and two FSVQ designs: the first FSVQ design used was a straightforward application of the design technique outlined previously and the second used the stochastic iteration next-state improvement algorithm of [53]. Observe that the next-state function improvement yields codes that perform better outside of the training sequence then do the ordinary FSVQ codes.

TABLE V
LPC VQ AND FSVQ WITH AND WITHOUT NEXT STATE FUNCTION IMPROVEMENT.

		VQ		FSVQ1		FSVQ2		
R	*r*	SNRin	SNRout	SNRin	SNRout	SNRin	SNRout	K
1	.008	3.7	2.9					
2	.016	6.1	5.2	7.2	4.3	7.5	6.1	16
3	.023	7.3	6.2	8.4	5.9	9.0	7.5	16
4	.031	8.8	7.9	9.5	7.8	9.6	8.7	4
5	.039	9.7	8.8	10.6	8.9	10.7	9.3	4
6	.047	10.5	9.5					
7	.055	11.6	10.1					
8	.062	12.6	10.7					

Signal to Noise Ratios inside training sequence (SNRin) of 5000 vectors of 128 samples each, Signal to Noise Ratios outside training sequence (SNRout) of 600 vectors of 128 samples each: memoryless VQ, omniscient FSVQ design (FSVQ1), and for omnisicient FSVQ design with next-state function improvement (FSVQ2). *K* = number of states in FSVQ, *R* = rate in bits/vector, *r* = rate in bits/sample. Itakura-Saito distortion measure.

TABLE VI
ADAPTIVE VPQ.

	VPQ	
k	SNRin	SNRout
1	4.12	4.34
2	7.47	7.17
3	8.10	7.67
4	8.87	8.30

Signal to Noise Ratios inside training sequence (SNRin) of 5000 vectors, and Signal to Noise Ratios in test sequence (SNRout) of 600 vectors, rate = 1.023 bits/sample.

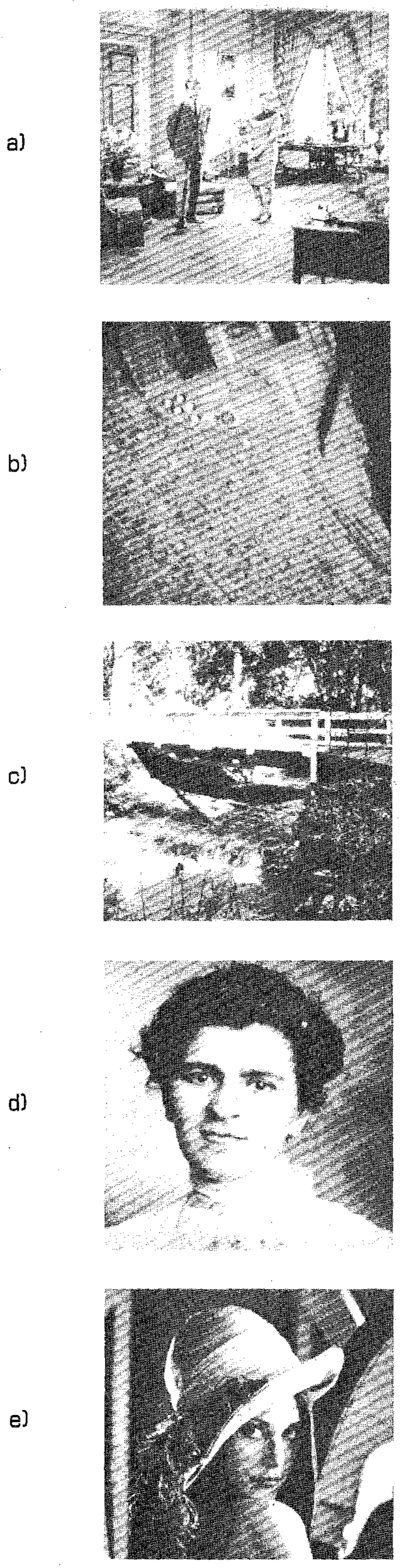


Figure 16. Image Training Sequence. The training sequence consisted of the sequence of 3 × 4 subblocks of the five 256 × 256 images shown.

Gain/shape VQ's for this application are developed in [16] and [36]. Tree-searched LPC VQ is considered for binary and nonbinary trees in combination with gain/shape codes in [16] and [10].

Adaptive coding

Table VI presents the results of a simple example of an adaptive VQ, here consisting of an LPC VQ with 8 codewords every 128 samples combined with VPQs of dimensions 1–4. Each of the 8 VPQs is designed for the subsequence of training vectors mapping into the corresponding LPC VQ model [47]. The rate of this system is $1 + 3/128 = 1.023$ bits/sample. The performance is significantly worse than the 10 dB achieved by a hybrid scalar trellis encoder of the same rate [52], but it improves on the nonadaptive VPQ by about ¾ dB. Adaptive vector quantizers are still quite new, however, and relatively little work on the wide variety of possible systems has yet been done.

Image coding

In 1980–1982 four separate groups developed successful applications of VQ techniques to image coding [61, 62, 63, 64, 65, 66, 67, 37]. The only real difference from waveform coding is that now the VQ operates on small rectangular blocks of from 9 to 16 pixels, that is, the vectors are really 2-dimensional subblocks of images, typically squares with 3 or 4 pixels on a side or 3 by 4 rectangles. We here consider both the basic technique and one variation. We consider only small codebooks of 6 bits per 4×3 block of 12 pixels for purposes of demonstration. Better quality pictures could be obtained at the same rate of ½ bit per pixel by using larger block sizes and hence larger rates of, say, 8 to 10 bits per block. Better quality could also likely be achieved with more complicated distortion measures than the simple squared error used.

Fig. 16 gives the training sequence of five images. Fig. 17a shows a small portion of the fifth image, an eye, magnified. Fig. 17b is a picture of the $2^6 = 64$ codewords. Fig. 17c shows the decoded eye. Fig. 18 shows the original, decoded image, and error image for the complete picture. The error image is useful for highlighting the problems encountered with the ordinary memoryless VQ. In particular, edges are poorly reproduced and the codeword edges make the picture appear "blocky." This problem was attacked by Ramamurthi and Gersho [62, 67] by constructing segmented (or union or composite) codes—separate codebooks for the edge information and the texture information where a simple classifier was used to distinguish the two in design. In [37] a feedback vector quantizer was developed by using a separating mean VQ with a predictive scalar quantizer to track the mean. Fig. 19 shows the original eye, ordinary VQ, and the feedback VQ. The improved ability to track edges is clearly discernible. Fig. 20 shows the full decoded image for feedback VQ together with the error pattern.

Although image coding using VQ is still in its infancy,

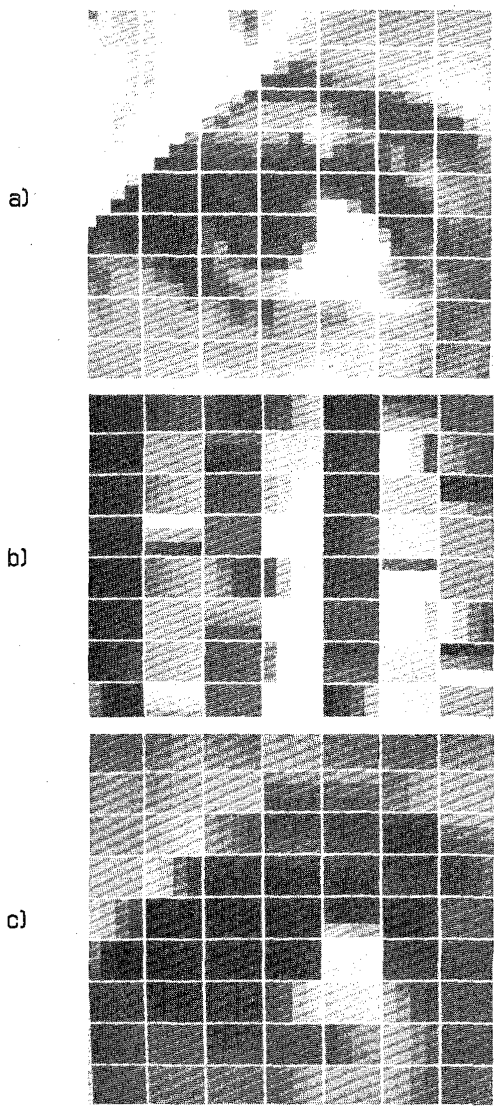


Figure 17. Basic Image VQ Example at ½ bit per pixel. (a) Original Eye Magnified (b) 6 bit codebook VQ codebook for 4 × 3 blocks (c) Decoded Image.

these preliminary experiments using only fairly simple memoryless and feedback VQ techniques with small codebooks demonstrate that the general approach holds considerable promise for such applications.

COMMENTS

We have described Lloyd's basic iterative algorithm and how it can be used to improve the performance of a variety of vector quantization systems, ranging from the fundamental memoryless full search VQ that serves as the basic model for data compression in information theory to a variety of feedback and adaptive systems that can be viewed as vector extensions of popular scalar compression systems. By a variety of examples of systems and code design simulations we have tried to illustrate some of the tradeoffs among performance, rate, complexity, and storage for these codes.

The basic structure of all of the VQ systems is well suited to VLSI implementation: a minimum distortion search algorithm on a chip communicating with off-board storage for codebooks and next-state transition functions. As new and better design algorithms are developed, the chips can be updated by simply reburning the codebook and transition ROM's.

The basic approach can also be incorporated into the design of some traditional scalar data compression schemes, an approach which Gersho calls "imbedded

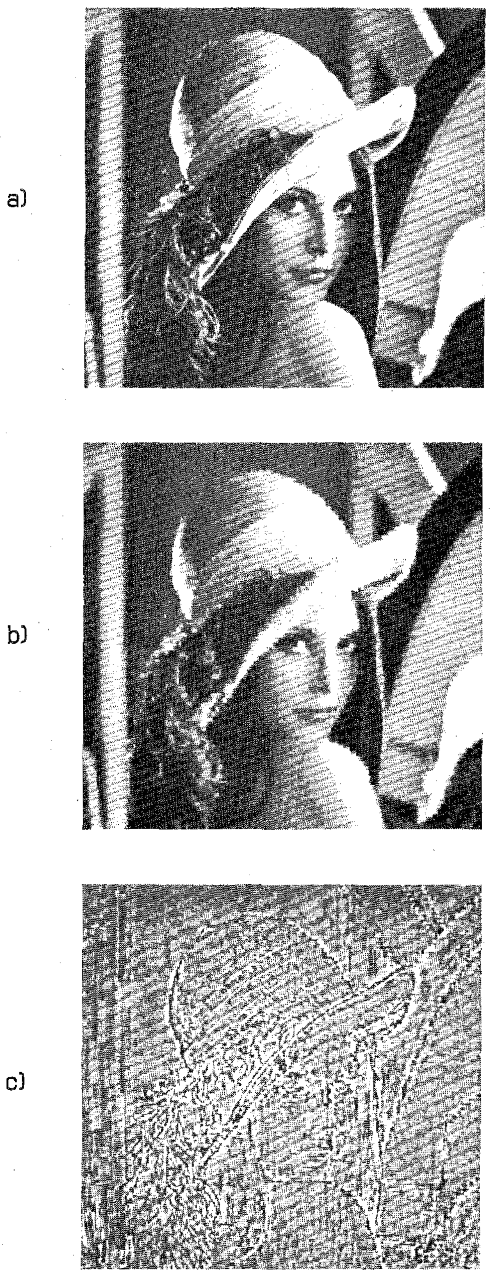
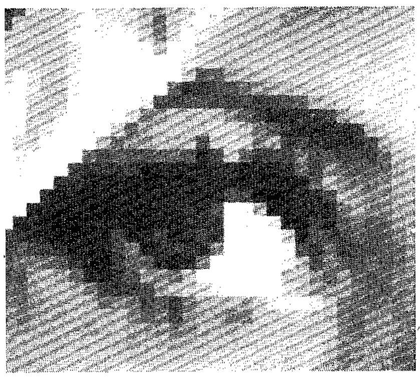
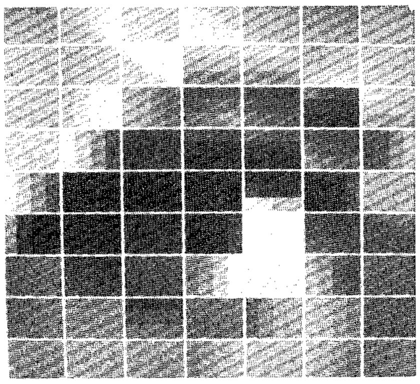


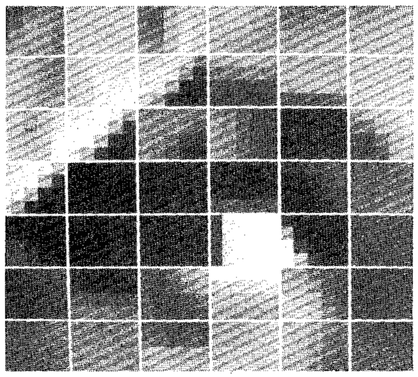
Figure 18. Full Image for Basic Example (a) Original (b) Decoded Image (c) Error Image.



a)

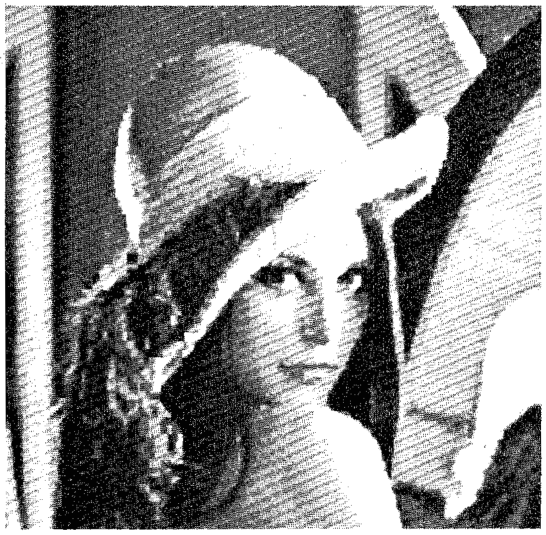


b)

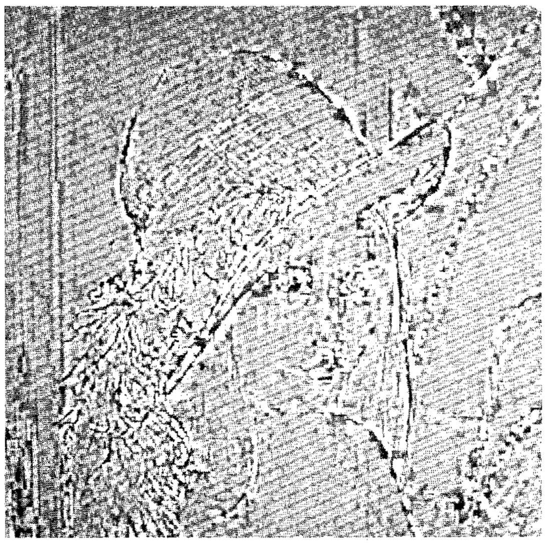


c)

Figure 19. VQ vs. Separating Mean VQ at Rate ½ bit per pixel (a) Original Eye Magnified (b) VQ Decoded Image (c) Separating Mean VQ with DPCM Mean Coding Decoded Image.



a)



b)

Figure 20. Full Image for Separating Mean Example (a) Decoded Image using Separating Mean VQ with DPCM Mean Coding (b) Error Image.

VQ" [11]. Such schemes typically enforce additional structure on the code such as preprocessing, transforming, splitting into subbands, and scalar quantization, however, and hence the algorithms may not have the freedom to do as well as the more unconstrained structures considered here. Even if the traditional schemes prove more useful because of existing DSP chips or intuitive variations well matched to particular data sources, the vector quantization systems can prove a useful benchmark for comparison.

Recently VQ has also been successfully used in isolated word recognition systems without dynamic time warping by using either separate codebooks for each utterance or by mapping trajectories through one or more codebooks [68,69,70,71,55,72]. Vector quantization has also been used as a front end acoustic processor to isolated utterance and continuous speech recognition systems which then do approximately maximum likelihood linguistic decoding based on probabilities estimated using "hidden Markov" models for the VQ output data. [73,74,75].

Variations of the basic VQ design algorithm have been tried for several distortion measures, including the squared error, weighted squared error, the Itakura-Saito distortion, and an (arithmetic) segmented signal to noise ratio. (See, e.g., [30,45,46]). Other distortion measures are currently under study.

The algorithm has not yet been extended to some of the more complicated distortion measures implicit in noise masking techniques for enhancing the subjective performance of scalar quantization speech coding systems. Whether scalar systems designed by sophisticated techniques matched to subjective distortion measures will sound or look better than vector systems designed for mathematically tractable distortion measures remains to be seen. Whenever the subjective distortion measures can be quantified and a means found to compute centroids, however, the vector systems will yield better quantitative performance. Since the centroid computation is only done in design and not in implementation, it can be quite complicated and still yield useful results.

The generalized Lloyd algorithm is essentially a clustering algorithm and we have attempted to demonstrate its applicability to the design of a variety of data compression systems. Other clustering algorithms may yield better codes in some applications. For example, Freeman [76] proposed a design algorithm for scalar trellis encoding systems using the squared error distortion measure which replaced the Lloyd procedure by a conjugate gradient procedure for minimizing the average distortion for a long training sequence. He found that for a memoryless Gaussian source the resulting codes were superior to those obtained by the Lloyd procedure. It would be interesting to characterize the reasons for this superiority, e.g., the procedure may find a better local minimum or it may simply be numerically better suited for finding a continuous local minimum on a digital computer. It would also be interesting to consider variations of this approach for the design of some of the other systems considered here.

A survey article with many topics cannot provide complete descriptions or exhaustive studies of any of systems sketched. It is hoped, however, that these examples impart the flavor of vector quantizer design algorithms and that they may interest some readers to further delve into the recent and current work in the area.

ACKNOWLEDGMENT

The author gratefully acknowledges the many helpful comments from students and colleagues that aided the preparation of this paper.

Portions of the research described here were supported by the Army Research Office, the Air Force Office of Scientific Research, the National Science Foundation, the John Simon Guggenheim Memorial Foundation, and the Joint Services Electronics Program at Stanford University.

REFERENCES

[1] Davisson, L. D. and Gray, R. M., *Data Compression,* Dowden, Hutchinson, & Ross, Inc., Stroudsbug, PA (1976). Benchmark Papers in Electrical Engineering and Computer Science, Volume 14.

[2] Jayant, N. S., Editor, *Waveform coding quantization and Coding,* IEEE Press, NY (1976).

[3] Jayant, N. S. and Noll, P., *Digital Coding of Waveforms,* Prentice–Hall, Englewood Cliffs, NJ (1984).

[4] Shannon, C. E., "A mathematical theory of communication," *Bell Systems Technical Journal* 27 pp. 379–423, 623–656 (1948).

[5] Shannon, C. E., "Coding theorems for a discrete source with a fidelity criterion," *IRE National Convention Record, Part 4,* pp. 142–163 (1959).

[6] Gallager, R. G., *Information theory and reliable communication,* John Wiley & Sons, NY (1968).

[7] Berger, T., *Rate Distortion Theory,* Prentice-Hall Inc., Englewood Cliffs, NJ (1971).

[8] Viterbi, A. J. and Omura, J. K., *Principles of Digital Communication and Coding,* McGraw-Hill Book Company, New York (1979).

[9] Lloyd, S. P., *Least squares quantization in PCM,* Bell Laboratories Technical Note (1957). (Published in the March 1982 special issue on quantization).

[10] Wong, D., Juang, B.-H., and Gray, A. H., Jr., "An 800 bit/s vector quantization LPC vocoder," *IEEE Transactions on Acoustics Speech and Signal Processing* ASSP-30 pp. 770–779 (October 1982).

[11] Gersho, A. and Cuperman, V., "Vector Quantization: A pattern-matching technique for speech coding," *IEEE Communications Magazine,* (December 1983).

[12] Itakura, F. and Saito, S., "Analysis synthesis telephony based on the maximum liklihood method," *Proceedings of the 6th International Congress of Acoustics,* pp. C-17–C-20 (August 1968).

[13] Kullback, S., *Information Theory and Statistics,* Dover, New York (1969).

[14] Gray, R. M., Gray, A. H., Jr., Rebolledo, G., and Shore, J. E., "Rate distortion speech coding with a minimum discrimination information distortion measure," *IEEE Transactions on Information Theory* IT-27 (6) pp. 708–721 (Nov. 1981).

[15] Shore, J. E. and Gray, R. M., "Minimum-cross-entropy pattern classification and cluster analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-4 pp. 11–17 (Jan. 1982).

[16] Buzo, A., Gray, A. H., Jr., and Gray, R. M., and Markel, J. D., "Speech coding based upon vector quantization," *IEEE Transactions on Acoustics Speech and Signal Processing* ASSP-28 pp. 562–574. (October 1980).

[17] Gray, R. M., Buzo, A., Gray, A. H., Jr., and Matsuyama, Y., "Distortion measures for speech processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* ASSP-28 pp. 367–376 (August 1980).

[18] Gray, R. M., Kieffer, J. C., and Linde, Y., "Locally opti-

mal block quantizer design," *Information and Control* 45 pp. 178–198 (May 1980).

[19] Gray, R. M. and Kieffer, J. C., "Asymptotically mean stationary measures," *Annals of Probability* 8 pp. 962–973 (Oct. 1980).

[20] Kieffer, J. C. and Rahe, M., "Markov channels are asymptotically mean stationary," *Siam Journal of Mathematical Analysis* 12 pp. 293–305 (1980).

[21] Fontana, R. J., Gray, R. M., and Kieffer, J. C., "Asymptotically mean stationary channels," *IEEE Transactions on Information Theory* IT-27 pp. 308–316 (May 1981).

[22] Kieffer, J. C., "Stochastic stability for feedback quantization schemes," *IEEE Transactions on Information Theory* IT-28 pp. 248–254 (March 1982).

[23] Sabin, M. J. and Gray, R. M., *Asymptotic properties of the generalized Lloyd algorithm,* Submitted for publication 1983.

[24] Gersho, A., "On the structure of vector quantizers," *IEEE Transactions on Information Theory* IT-28 pp. 157–166 (March 1982).

[25] Linde, Y., Buzo, A., and Gray, R. M., "An algorithm for vector quantizer design," *IEEE Transactions on Communications* COM-28 pp. 84–95 (January 1980).

[26] MacQueen, J., "Some methods for classification and analysis of multivariate observations," *Proc. of the Fifth Berkeley Symposium on Math. Stat. and Prob.* 1 pp. 281–296 (1967).

[27] Diday, E. and Simon, J. C., "Clustering analysis," in *Digital Pattern Recognition,* ed. K. S. Fu, Springer-Verlag, NY (1976).

[28] Chen, D. T. S., "On two or more dimensional optimum quantizers," *Proceedings, 1977 International Conference on Acoustics, Speech, and Signal Processing,* pp. 640–643 (1977).

[29] Adoul, J.-P., Morissette, S., and Rudko, M., "Bit-rate-halving algorithm for PCM-encoded speech using a new bidimensional data compression scheme," *Record of the 1979 IEEE International Conference on Acoustics Speech and Signal Processing,* pp. 432–435 (April 1979).

[30] Gray, R. M. and Karnin, E., "Multiple local optima in vector quantizers," *IEEE Transactions on Information Theory* IT-28 pp. 256–261 (March 1982).

[31] Abut, H., Gray, R. M., and Rebolledo, G., "Vector quantization of speech and speech-like waveforms," *IEEE Transactions on Acoustics Speech and Signal Processing* ASSP-30 pp. 423–435 (June 1982).

[32] Adoul, J.-P., Debray, J.-L., and Dalle, D., "Spectral distance measure applied to the optimum design of DPCM coders with L predictors," *Proceedings of the 1980 IEEE International Conference on Acoustics Speech and Signal Processing,* pp. 512–515 (April 1980).

[33] Gersho, A. and Cheng, D., "Fast nearest neighbor search for nonstructured Euclidean codes," *Abstracts of the 1983 IEEE International Symposium on Information Theory,* p. 88 (September 1983).

[34] Juang, B.-H. and Gray, A. H., Jr., "Multiple stage vector quantization for speech coding," *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing* 1 pp. 597–600 (April 1982).

[35] Sabin, M. J. and Gray, R. M., "Product code vector quantizers for speech waveform coding," *Conference Record Globecom '82,* pp. 1087–1091 (December 1982).

[36] Sabin, M. J. and Gray, R. M., *Product code vector quantizers for waveform and voice coding, IEEE Trans. ASSP,* to appear, (April 1984).

[37] Baker, R. L. and Gray, R. M., "Differential vector quantization of achromatic imagery," *Proceedings of the International Picture Coding Symposium,* (March 1983).

[38] Gersho, A., "Asymptotically optimal block quantization," *IEEE Transactions on Information Theory* IT-25 pp. 373–380 (July 1979).

[39] Conway, J. H. and Sloane, N. J. A., "Voronoi regions of lattices, second moments of polytopes, and quantization," *IEEE Transactions on Information Theory* IT-28 pp. 211–226 (March 1982).

[40] Conway, J. H. and Sloane, N. J. A., "Fast quantizing and decoding algorithms for lattice quantizers and codes," *IEEE Transactions on Information Theory* IT-28 pp. 227–232 (March 1982).

[41] Conway, J. H. and Sloane, N. J. A., "On the Voronoi regions of certain lattices," *SIAM Journal of Alg. Disc. Math.,* (1983). in press

[42] Barnes, E. S. and Sloane, N. J. A., "The optimal lattice quantizer in three dimensions," *SIAM Journal of Alg. Disc. Math.* 4 pp. 30–41 (1983).

[43] Foster, J., Newkirk, J., and Gray, R. M., *VLSI implementation of a finite-state vector quantization waveform encoder,* submitted for publication 1983.

[44] Gaarder, N. T. and Slepian, D., "On optimal finite-state digital transmission systems," *IEEE Transactions on Information Theory* IT-28 pp. 167–186 (March 1982).

[45] Cuperman, V. and Gersho, A., "Adaptive differential vector coding of speech," *Conference Record, GlobeCom 82,* pp. 1092–1096 (December 1982).

[46] Cuperman, V. and Gersho, A., *Vector predictive coding of speech at 16 Kb/s,* Submitted for possible publication 1983.

[47] Chang, P. C., Ph.D. Research, Stanford University 1983.

[48] Gibson, J. D., Jones, S. K., and Melsa, J. L., "Sequentially adaptive prediction and coding of speech signals," *IEEE Transactions on Communications* COM-22 pp. 1789–1797 (November 1974).

[49] Dunn, J. G., "An experimental 9600-bit/s voice digitizer employing adaptive prediction," *IEEE Transactions on Communication Technology* COM-19 pp. 1021–1032 (December 1971).

[50] Foster, J. and Gray, R. M., "Finite-state vector quantization," *Abstracts of the 1982 International Sym-*

posium on Information Theory, (June 1982).
[51] Foster, J., Gray, R. M., and Ostendouf, M., *Finite-state vector quantization for waveform coding,* IEEE Trans. Info. Theory, to appear.
[52] Stewart, L. C., Gray, R. M., and Linde, Y., "The design of trellis waveform coders," *IEEE Transactions on Communications* COM-30 pp. 702–710 (April 1982).
[53] Ostendorf, M. and Gray, R. M., *An algorithm for the design of labeled-transition finite-state vector quantizers,* submitted for publication 1983.
[54] Fehn, H. G. and Noll, P., "Multipath search coding of stationary signals with applications to speech," *IEEE Transactions on Communications* COM-30 pp. 687–701 (April 1982).
[55] Shore, J. E. and Burton, D. K., "Discrete utterance speech recognition without time alignment," *IEEE Transactions on Information Theory* IT-29 pp. 473–491 (July 1983).
[56] Rebolledo, G., Gray, R. M., and Burg, J. P., "A multirate voice digitizer based upon vector quantization," *IEEE Transactions on Communications* COM-30 pp. 721–727 (April 1982).
[57] Adoul, J.-P. and Mabilleau, P., "4800 bps RELP vocoder using vector quantization for both filter and residual representation," *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing* 1 p. 601 (April 1982).
[58] Heron, C. D., Crochiere, R. E., and Cox, R. V., "A 32-band subband/transform coder incorporating vector quantization for dynamic bit allocation," *Proceedings ICASSP,* pp. 1276–1279 (April 1983).
[59] Gray, R. M. and Linde, Y., "Vector quantizers and predictive quantizers for Gauss-Markov sources," *IEEE Transactions on Communications* COM-30 pp. 381–389 (Feb. 1982).
[60] Arnstein, D. S., "Quantization error in predictive coders," *IEEE Transactions on Communications* COM-23 pp. 423–429 (April 1975).
[61] Yamada, Y., Fujita, K., and Tazaki, S., "Vector quantization of video signals," *Proceedings of Annual Conference of IECE,* p. 1031 (1980).
[62] Gersho, A. and Ramamurthi, B., "Image coding using vector quantization," *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing* 1 pp. 428–431 (April 1982).
[63] Baker, R. L. and Gray, R. M., "Image compression using non-adaptive spatial vector quantization," *Conference Record of the Sixteenth Asilomar Conference on Circuits Systems and Computers,* (October 1982).
[64] Murakami, T., Asai, K., and Yamazaki, E., "Vector quantizer of video signals," *Electronic Letters 7* pp. 1005–1006 (Nov. 1982).
[65] Yamada, Y. and Tazaki, S., "Vector quantizer design for video signals," *IECE Transactions* J66-B pp. 965–972 (1983). (in Japanese)
[66] Yamada, Y. and Tazaki, S., "A method for constructing successive approximation vector quantizers for video signals," *Proceedings of the Annual Conference of the Institute of Television Engineers of Japan,* pp. 6–2 (1983).
[67] Ramamurthi, B. and Gersho, A., "Image coding using segmented codebooks," *Proceedings International Picture Coding Symposium,* (Mar. 1983).
[68] Hamabe, R., Yamada, Y., Murata, M., and Namekawa, T., "A speech recognition system using inverse filter matching technique," *Proceedings of the Ann. Conf. Inst. of Television Engineers,* (June 1981). (in Japanese)
[69] Shore, J. E. and Burton, D. K., "Discrete utterance speech recognition without time alignment," *Proceedings 1982 IEEE International Conference on Acoustics Speech and Signal Processing,* p. 907 (May 1982).
[70] Martinez, H. G., Riviera, C., and Buzo, A., "Discrete utterance recognition based upon source coding techniques," *Proceedings IEEE International Conference on Acoustics Speech and Signal Processing,* pp. 539–542 (May 1982).
[71] Rabiner, L., Levinson, S. E., and Sondhi, M. M., "On the application of vector quantization and hidden Markov models to speaker-independent isolated word recognition," *Bell System Technical Journal* 62 pp. 1075–1106 (April 1983).
[72] Shore, J. E., Burton, D., and Buck, J., "A generalization of isolated word recognition using vector quantization," *Proceedings 1983 International Conference on Acoustics Speech and Signal Processing,* pp. 1021–1024 (April 1983).
[73] Jelinek, F., Mercer, R. L., and Bahl, L. R., "Continuous speech recognition: statistical methods," in *Handbook of Statistics,* Vol. 2, P. R. Khrishaieh and L. N. Kanal, eds., North-Holland, pp. 549–573. (1982).
[74] Billi, R., "Vector quantization and Markov models applied to speech recognition," *Proc. ICASSP 82,* pp. 574–577, Paris (May 1982).
[75] Rabiner, L. R., Levinson, S. E., and Sondhi, M. M., "On the application of vector quantization and hidden Markov models to speaker-independent isolated word recognition," *BSTJ,* Vol. 62, pp. 1075–1105, (April 1983).
[76] Freeman, G. H., "The design of time-invariant trellis source codes," *Abstracts of the 1983 IEEE International Symposium on Information Theory,* pp. 42–43 (September 1983).
[77] Haoui, A. and Messerschmidt, D., "Predictive Vector Quantization," Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (1984).

Robert M. Gray was born in San Diego, CA, on November 1, 1943. He received the B.S. and M.S. degrees from M.I.T. in 1966 and the Ph.D. degree from U.S.C. in 1969, all in Electrical Engineering. Since 1969 he has been with the Information Systems Laboratory and the Electrical Engineering Department of Stanford University, CA, where he is currently a Professor engaged in teaching and research in communication and information theory with an emphasis on data compression. He was Associate Editor (1977–1980) and Editor (1980–1983) of the *IEEE Transactions on Information Theory* and was a member of the IEEE Information Theory Group Board of Governors (1974–1980). He was corecipient with Lee D. Davisson of the 1976 IEEE Information Theory Group Paper Award. He has been a fellow of the Japan Society for the Promotion of Science (1981) and the John Simon Guggenheim Memorial Foundation (1981–1982). He is a fellow of the IEEE and a member of Sigma Xi, Eta Kappa Nu, SIAM, IMS, AAAS, and the Societé des Ingenieurs et Scientifiques de France. He holds an Advanced Class Amateur Radio License (KB6XQ).

Notes added in proof: A similar design technique for FSVQ was independently developed by Haoui and Messerschmidt [77]. It should be pointed out that the FSVQ design algorithm described here is incomplete in that it does not describe the methods used to avoid non-communicating collection of states and wasted states. These issues are discussed in [51].