N89-22340

VECTOR QUANTIZATION

Robert M. Gray
Information Systems Laboratory

ABSTRACT

During the past ten years Vector Quantization (VQ) has developed from a theoretical possibility promised by Shannon's source coding theorems into a powerful and competitive technique for speech and image coding and compression at medium to low bit rates. In this survey, the basic ideas behind the design of vector quantizers are sketched and some comments made on the state-of-the-art and current research efforts.

INTRODUCTION

VQ can be thought of as the vector extension of Pulse Coded Modulation (PCM) wherein real vectors instead of real scalars are converted into digital representations which in turn can be used to produce a reproduction of the original signal. The goal, of course, is to produce a digital representation of the signal which can be communicated on a digital communication channel or stored in a digital medium. Representing analog data digitally introduces distortion, and hence a major design goal is to minimize the distortion given constraints on communication or storage capacity and complexity. The vectors to be digitized may be a collection of consecutive samples from a continuous waveform, rectangular subblocks of an image intensity or density, three dimensional vectors consisting of, say, two-by-two squares of pixels three frames deep (or twelve pixels in the vector), or they may be feature or parameter vectors extracted from the data which represent its important attributes, such as Fourier transformed vectors or the Linear Predictive Coded (LPC) representation of a speech signals.

Although Shannon's source coding theorems imply that performance improvement can always be obtained by coding vectors instead of scalars [1,2,3,55,62], the most popular systems for analog-to-digital conversion and data compression perform the quantization operation only on scalars, although they often effectively operate on vectors by imbedding the quantizer in a feedback loop (as in predictive quantization) or by first performing a linear transform on the data (as in transform coding). While such systems have the advantage of simplicity, they are necessarily suboptimal in the Shannon sense. Furthermore, the definition of "simplicity" has been enormously extended with modern circuit design and implementation techniques: DSP chips and VLSI have rendered feasible algorithms that were considered absurdly complicated only ten years ago.

In addition to the complexity issue, another impediment to the use of VQ systems has been the lack of design algorithms. Unlike the dual problem of channel coding or error control coding, quantization is fundamentally nonlinear and the algebraic approaches successful in error correction are of little help in the digitization problem. Since the middle of the last decade, a variety of design techniques and tricks have been developed for VQ and real time hardware has been designed to perform sophisticated variations of these systems. This paper presents a brief overview of the fundamental design principles of the basic VQ structure and its variations. Deeper discussions of many of the issues and systems may be found in tutorial articles[4,5,6]. A thorough development of VQ systems may be found[7].

VQ is a form of "lossy" data compression in contrast to "lossless" data compression or noiseless coding. Noiseless codes are perfectly invertible and necessarily variable length codes. The most popular noiseless coding algorithms are Huffman coding, Rice codes, Lempel-Ziv Codes, and arithmetic codes[2,55-61]. These codes are in common use, especially for the compression of computer programs and data files which cannot tolerate errors. Noiseless compression is usually

required in such situations where the compression system must be designed without any knowledge of the structure or end use of the original digital signal. On the other hand, any system which begins with an analog signal (such as microphone, camera, or analog sensor output) and produces a digital output is necessarily a lossy code as a continuous voltage cannot be reproduced perfectly from a digital representation. Given that all such analog-to-digital conversion systems are lossy, the goal of any such system is to provide the best quality (minimum loss) within the constraints of the system. As unpleasant as purposefully introducing distortion into a representation might sound to a user, it is preferable to the potential large insertion of uncontrolled distortion or the complete loss of the data caused by overwhelming available digital communication or storage capacity. In other words, if you are generating gigabits but the available communication channel only takes megabits, then you either compress to the best acceptable quality or you may get no useful data at all. VQ is an approach to minimizing the loss for a given communication or storage rate. It is based on the Shannon theory approach of defining and minimizing an objective distortion criterion for a given code rate. The minimization is accomplished using algorithms developed in communications, statistics, and cluster analysis. Next two sections summarize the basic approach.

MATHEMATICAL MODELS

The mathematical model for an analog-to-digital conversion system or for a data compression system is a source code subject to a fidelity criterion. A source $\{X(n);n=1,2,...\}$ is a discrete time signal which in general is vector-valued. Let A denote the alphabet or collection of possible values of X(n)., for example, A may be k dimensional Euclidean vector space. For convenience we assume that the signal is a statistically "nice" process (e.g., the law of large numbers holds). The basic results extend to more general processes.

A code in the general sense is a mapping of the input sequence $\{X(n)\}$

into a binary sequence (the encoder) together with a mapping of the binary sequence into a reproduction sequence {Y(n)} (the decoder). (We assume the encoded sequence is binary for convenience, in general it need only be from a finite alphabet.) The rate (or resolution) R of the code is the number of bits or binary symbols transmitted or stored per source symbol. In general this rate can be fractional, although in some examples it is useful to focus on integral values. A block source code is a code where each input block or vector (X(lk), X(lk+1),...,X((l+1)k-1)) is mapped into its binary code word in a way that does not depend on past or future actions of the encoder. The decoder is required to act in a similar fashion independently of past and future vectors. A block source code is also called a (memoryless) block quantizer or vector quantizer. The qualifier "memoryless" reflects the fact that such codes operate on vectors in a memoryless fashion (although the clearly have memory with respect to the individual symbols within the vectors). We will consider codes that have memory, but the focus of Shannon theory is on memoryless codes.

To measure the performance of a code, we assume a non-negative distortion measure d(x,y) which measures the cost of reproducing any x in A by some y in a reproduction alphabet B, which may or may not be the same as A. The performance is measured by an average distortion, where the average may be a long term time average or an ensemble or probabilistic average. For convenience we represent both by

$$\Delta_N = E\left[\frac{1}{N}\sum_{i=1}^{N} d(X(i), Y(i))\right],$$

where the expectation E can mean either a probabilistic average or a time average (which can be viewed as a special case of a probabilistic average in which every sample vector in a training sequence of length L has probability 1/L. Ideally a distortion measure should be analytically tractable, computable, and subjectively meaningful. In practice, these attributes must be balanced and a variety of choices exist.

Shannon's converse coding theorem and its generalizations imply that for any code for which these definitions make sense, Δ can be no smaller than the distortion-rate function D(R) of the source and distortion measure evaluated at rate R, a function defined by an information theoretic minimization which can be computed analytically or numerically or bounded for many interesting sources. Shannon also showed that performance arbitrarily near to D(R) could be achieved with block source codes, another name for VQ. Unfortunately, however, the proof of this result provided no indication of how to actually design a good code. This we explore shortly.

MEMORYLESS VECTOR QUANTIZATION

As described in the previous section, a memoryless vector quantizer or block quantizer is in the Shannon terminology a length k block source code subject to a fidelity criterion; that is, it is a pair of mappings, an encoder E which maps k-dimensional input vectors X into binary vectors which we denote by their equivalent decimal representation j = 1,2,...,M, and a decoder D which maps those binary vectors into reproduction vectors. For simplicity we assume that the binary vectors have dimension K and hence that the rate of the code is R = K/k bits per input symbol. To describe the operation of a block code define the code book C= {y (j) = D(j), j = 1,2,...,M} as the collection of all possible reproduction vectors, and the code partition P = { P(j);j=1,...,M}, where P(j) is the collection of all input vectors which are encoded into the binary vector j. The quantizer mapping Q(x)is defined as D(E(x)), that is, the overall action of the code. The term VQ is used to refer to combination of the encoder and decoder or, equivalently, the overall mapping Q.

The average distortion resulting from applying a vector quantizer to a source can be written using conditional averages as

$$\Delta = \sum_{j=1}^{M} E\ [d(X,Y(j))|X\epsilon P(j)]P(X\epsilon P(j))$$

Recall that the expectation and the probabilities may come either from a probabilistic model or (more commonly) from a training sequence of typical data. A vector quantizer is optimal if it yields the smallest possible Δ over any quantizer with the same dimension and resolution. The above representation easily yields two necessary conditions for a VQ to be optimal.

The Nearest Neighbor (Minimum Distortion) Condition

A necessary condition for a VQ to be optimal is that the encoder be optimal for the decoder. This is equivalent to the following: If the decoder yields a code book C, then the encoder must be a nearest neighbor or minimum distortion mapping that satisfies

$$E(x) = j \text{ only if } d(x,y(j)) \leq d(x,y(i)), \text{ all } i \neq j$$

Thus given a decoder or, equivalently, a code book, the optimal encoder is the one which searches the entire code book and selects the binary vector corresponding to the minimum distortion available reproduction vector.

The Centroid Condition

Define the centroid of a set S with respect to a distortion measure d and a probability distribution on X = (X(1), ... ,X(N)) by

$$\text{cent}(s) = \min_{y}^{-1} E[d(X,Y)|X\epsilon S]$$

A necessary condition for a VQ to be optimal is that the decoder be optimal for the encoder. This is equivalent to the following: If the encoder implies a partition {P(j)}, then the optimal decoder satisfies

$$D(j) = \text{cent}(P(j)); \quad \text{all } j.$$

For example, in the case of mean squared error, the centroid is simply a conditional mean given that the input was mapped into the binary vector j. This condition states that given an encoder, which can be considered as a partition of the input vector space, then the optimum decoder is the one which maps each received binary vector into the centroid of the region of the partition which is encoded into that binary vector. Note that unlike the encoder condition, this condition requires knowledge of the input signal distribution, knowledge that can come from a mathematical model or from a training sequence.

MEMORYLESS VQ DESIGN

Because of the nearest neighbor condition, a VQ is completely described by its code book. The two properties together provide a means of improving any given code book C:

The Lloyd Iteration

1. Given a code book C, form a nearest neighbor partition {P(j)}.

2. Given a partition {P(j)}, form a new code book C'= {cent(P(j)); j=1, ... , M}.

It is obvious that the above operation produces a new code book that is better than (at least no worse than) the original code book since each step can only improve performance. These properties were first observed for the mean squared error and scalar quantizers by Lukaszewicz and Steinhaus[8] and were independently found shortly thereafter by Lloyd[9], who developed an iterative algorithm for designing scalar quantizers with a mean squared error based on repeated use of the iteration. The basic idea extends immediately to vectors and is called the generalized Lloyd algorithm:

The Lloyd VQ Design Algorithm

0. Given an initial code book C(0) and a threshold ∂.
   Set $\Delta(0)$ = huge. Set k = 1.
1. Use the Lloyd iteration to produce a new code book C(k) from C(k-1). 2. Evaluate the distortion

$$\Delta_k = E(\min_y d(X,Y))$$

If

$$\frac{\Delta_{k-1} - \Delta_k}{\Delta_k} < \partial$$

quit. Otherwise replace k by k+1 and continue.

In most practical applications, one does not have a probability distribution, but does have a training sequence of data. In this case the algorithm can be run on the empirical distribution which assigns a probability of 1/L to each of L samples in the training sequence. In this case the expected distortion is replaced by a sample average.

Theorems can be proved to the effect that if the training sequence is long enough, the VQ designed will be close to that which would have been designed had the distribution been known[10,11].

This algorithm was developed in the statistical literature under the name of the k-means algorithm by MacQueen[12] and was first applied to vector quantization in the two dimensional case with a mean squared error by Chen[13]. The algorithm was extended to general vector quantization with a variety of distortion measures by Linde, Buzo, and Gray[14], who computed centroids for a variety of distortion measures and applied the algorithm to speech waveform and voice compression.

A remaining issue is how to design the initial code book, which is in itself a code design problem. We now next describe several such techniques. In fact, these techniques can be used as an alternative to the Lloyd algorithm for designing a complete code, but such code books can always be improved by subsequent application of the Lloyd algorithm.

Random Coding

Perhaps the simplest conceptual approach towards filling a code book of M code words is to randomly select the code words according to the source distribution, which can be viewed as a Monte Carlo code book design. The obvious variation when designing based on a training sequence is to simply select the first M training vectors as code words. If the data is highly correlated, it will likely produce a better code book if one takes, say, every Nth training vector. This technique has often been used in the pattern recognition literature and was used in the original development of the k-means technique by MacQueen[12]. One can be somewhat more sophisticated and randomly generate a code book using not the input distribution, but the distribution which solves the optimization problem defining Shannon's distortion-rate function. In fact, the Shannon source coding theorems imply that such a random selection will on the average yield a good

code. Unfortunately, the code book will have no useful structure and may turn out to be quite awful.

Observe that here "random coding" means only that the code book is selected at random: once selected it is used in the usual (nearest neighbor) deterministic way.

Pruning

A variation on the above use of a training sequence to populate a code is to form a code book recursively as follow: Put the first vector in the training sequence in the code book. Then compute the distortion between the next training vector and the first code word. If it is less than some threshold, continue. If it is greater than the threshold, add the new vector to the code book as a codeword. Continue in this fashion: With each new training vector, find the nearest neighbor in the code book. If the resulting distortion is not within some threshold, add the training vector to the code book. Continue in this fashion until the code book has enough words. This technique has been used in the statistical clustering literature[15].

Product Codes

In some cases a product code book may provide a good initial guess. For example, if one wishes to design a code book for a k-dimensional VQ with $M = 2^{kR}$ code words for some integer R, then one can use the product of k scalar quantizers with 2^R words each. Thus, if q(x) is a scalar quantizer, then Q(x(1), ... , x (k))= (q (x(0)), ... , q(x (k)), the cartesian product of the scalar quantizers, is a vector quantizer. This technique will not work if R is not an integer. In general other product structures can be used, e.g., one could first design a one dimensional quantizer q from scratch (perhaps using a uniform quantizer as an initial guess). One could then use (q(x (0) , q(x(1)) as an initial guess to design a good two-dimensional quantizer Q(x(0), x(1)). One could then initiate a three dimensional

VQ design with the product (q(x(0)), Q(x(1), x(2))) as an initial guess. One could continue in this way to construct higher dimensional quantizers until the final size is reached.

Splitting

Linde et al. introduced a technique that resembles the product code initialization in that it grows large code books from small ones, but differs in that it does not require an integral number of bits per symbol[14]. The basic idea is this: The globally optimal rate 0 code book of a training sequence is the centroid of the entire sequence. The one code word, say w(0), in this code book can be "split" into two code words, w(0) and w(0)+∂, where ∂ is a vector of small Euclidean norm. This new code book has two words and can be no worse than the previous code book since it contains the previous code book. The Lloyd algorithm can be run on this code book to produce a good resolution 1 code. When complete, the training sequence can be divided into two smaller training sequences (one for each of the two binary code words). For each of these sub-training sequences and the corresponding single reproduction vector we can repeat the design process: each of the code words in the new code book is split, forming an initial guess for a rate 2 bit code book, and the Lloyd iteration is run to convergence, producing a binary code book for the corresponding sub-training sequence. One continues in this manner, using a good rate r code book to form an initial rate r+1 code book by splitting. This algorithm provides a complete design technique from scratch on a training sequence based on a training sequence. Another approach to splitting is an application of the "greedy" decision tree design[23]: Instead of splitting every node in a given level or depth of the tree together, split one node at a time by only splitting that node contributing the largest distortion to the overall distortion. In other words, each time the "worst" node is split. If a particular branch reaches the final permitted depth of the tree, it is then no longer permitted to split. This technique was suggested for code design in[6] and again provides a means of building a code

book from scratch. As we shall see, these designs provide a useful tree structure to the VQ that can be exploited for fast encoding. Yet another alternative is to begin with, say, a random code. Each new training vector is then associated with one of the M current code words and the training vector is added to a corresponding cluster or group. The code word for that group is then replaced by the centroid of all training vectors in the group, including the new addition. This is in fact the way the original k-means algorithm operated[12].

Pairwise Nearest Neighbor Merging

An alternative scheme for producing an initial guess is to begin by considering every member of the training sequence as a cluster. At each step one chooses a pair of two clusters and merges them by grouping the vectors in each in a new common cluster and assigning the new cluster its centroid as a code word. One can choose the pair that provides the best possible change in average distortion or approximately so[63,64].

We close this section by observing that in place of the Lloyd algorithm, one can also iteratively design VQ code books by using standard gradient search algorithms[16].

CONSTRAINED MEMORYLESS QUANTIZERS

A serious problem with an ordinary VQ is that its complexity and memory grow exponentially with resolution. Although the general structure of such searches is amenable to implementation using VLSI systolic arrays[17,18], any reduction in search complexity permits better performance at a given resolution. In the special case of mean squared error fidelity criteria, there are a variety of tricks that can be used to reduce the complexity of full searching of arbitrary code books[19]. More generally, however, the easiest means of reducing search complexity is to impose additional structure on the code book in order to permit rapid searches for nearest neighbors or

almost-nearest neighbors. The resulting loss of optimality may be compensated by the reduced implementation complexity. This can provide higher quality for a given rate and complexity, e.g., by permitting larger vector dimensions. Variations of the Lloyd algorithm can be run in order to produce a good code having the desired structure. Some of the code structures that have been studied are mentioned below:

Tree-Searched VQ

In a tree-searched VQ the full search of available codewords is replaced by a suboptimal tree search. The codeword is selected by a sequence of binary decisions instead of a single large search. [20,54,21,22,19,63,64]. The advantage is that the complexity of the search will be linear in rate instead of exponential in rate. A disadvantage is that general algorithms providing good suboptimal searches for general codebooks are not known. A classical problem in computer science is to find fast nearest neighbor tree searches for unstructured code books. If, however, the code book is constructed with an eventual tree search in mind (a freedom not always possible in the computer science applications), then good design algorithms do exist, as considered next.

Tree-Structured VQ

A tree-structured vector quantizer (TSVQ) is an example of a tree-searched code where the codebook itself is forced to have a tree structure and hence the tree search algorithm is natural. In particular, the encoder makes R binary distortion computations and comparisons instead of a single search requiring $2^{(NR)}$ distortion computations. The first tree-search and tree-structured codes were designed by a variation of the splitting algorithm of VQ design: Use the splitting algorithm to design a complete code book and do not run the Lloyd algorithm on the final complete code book. Instead retain the entire tree used in the design, that is, all of the binary code

books and the order in which they are used. This provides the TSVQ which is encoded by finding the minimum distortion path through the R layers of the tree; that is, one makes a sequence of R minimum distortion comparisons using a sequence of binary codes. After the final selection one achieves a terminal node (leaf) of the tree, which corresponds to the final code word. The decoder need only have the final code book and is a table lookup as before. A TSVQ is suboptimal because of its constrained structure, but it has two important advantages: It has low complexity and a fast encoder in comparison with an ordinary VQ since it needs to make only R binary comparisons instead of 2^R comparisons for a code book of size 2^R. In addition, the code has a nice successive approximation property in that each additional bit in the code word provides increased fidelity of the reproduction. This is a useful property, for example, in systems where the rate may be changed due to available communication channel capacity or where the communication link is slow and it is useful to get ever better quality reproduction as additional bits arrive.

Multistep VQ

Multistep VQ is a special case of tree-searched VQ where the same code book is used at all nodes at a given depth of the tree. This is usually accomplished by coding an error or residual produced by encoding the original signal using the code books previously encountered in the tree. For example, the first level does a coarse quantization of the input vector. The second level quantizes the difference of the original vector and the first level reproduction. The second level reproduction is formed by adding the first level reproduction to the second level error reproduction. The third level then quantizes the error resulting in the second level reproduction, and so on[24].

Hierarchical VQ

A VQ encoder can be constructed with a hierarchical structure, either by using long term parameter estimates to choose short term code books[25] or by using a sequence of ever longer dimension table lookups with a fixed code book size[26].

Product Codes

As previously described, a VQ code book can be decomposed into a cartesian product of smaller code books, a typical decomposition being into code books for gain (energy, mean, residual energy) and shape. Separate attributes are coded separately, but the coding is interdependent because the selection of specific code books and the distortion measure can depend on previously chosen code words[20,27,28].

Transform VQ

As a variation on a traditional scalar quantization technique, one can take a transform of a large window of data and then use VQ on the resulting transformed vectors[29]. A similar (and older) technique is to filter the input process into subbands and use the vector generalization of subband coding by applying VQ to the separate outputs[30,31]. By generalizing the notion of a transform to include any preprocessing of the data to enhance important features or well match human senses, good codes can often be found by combining linear filtering (possibly two dimensional) and VQ[32].

RECURSIVE QUANTIZERS

A VQ can be made to have memory by having a different code book for each state of the code. The encoder is given an input and a state and produces both an index of a code word in the state code book and a

next state. The decoder tracks the state in order to decode using the correct code book. Two forms of recursive VQ have been extensively studied in recent years. The first is predictive VQ[33,16], a natural extension of predictive quantization (or DPCM) to vectors wherein a linear vector predictor is formed based on the decoded reproduction and subtracted from the input. The resulting residual is then put into a VQ. The second approach is finite state VQ (FSVQ), which is a form of switched VQ consisting of a finite collection of codebooks (each associated with an encoder state) together with a next-state rule which determines the next codebook from the current one and the encoded word. Finite state codes were introduced by Shannon[62] and design algorithms for vector quantization were developed in [34,35,36,37]. Given the next-state rule, the design goal is to have an intelligent rule for selecting future code books based on past choices. These designs can be based on arguments from prediction theory, stochastic automata theory, or classification techniques. The latter approach, pioneered in [36,37,38], uses a simple classifier to detect important local image attributes such as background or edge detection and orientation. This classifier is used to divide a training sequence into vectors (blocks) which follow each occurrence of each class type. The sub-training sequences are then used to design memoryless VQs. An FSVQ is then constructed by classifying the decoder reproduction rather than the actual input, thus closing the loop and permitting the decoder to track the encoder state from a knowledge of the initial state and the received code words. These codes have produced excellent quality monochromatic images at 1/3 to 1/2 bit per pixel and real-time hardware implementations have been designed[39].

A general theoretical treatment of recursive VQ may be found in a recent book by Gabor and Gyorfi[40].

TRELLIS ENCODERS

If the decoder is a recursive VQ, then superior performance can be

obtained by replacing the encoder by a trellis encoder (or delayed decision or multipath encoder). This produces a trellis encoding system which is designed using a variation of the Lloyd algorithm and such systems have proved quite effective in waveform coding applications[41,42,43]. As with VQ, one can also design trellis codes by using gradient search techniques instead of the Lloyd algorithm[44,45]

MODEL VQ

Perhaps the most successful application of VQ to date has been that of LPC VQ, a form of model VQ wherein one codes an autoregressive (all-pole) model of a window of speech instead of the waveform itself[14,20]. Here a complicated distortion measure such as the Itakura-Saito distortion is used, but the Lloyd algorithm still works easily since the distortion can still be written as an inner product and a simple centroid computed. By grouping several LPC vectors together, one can achieve even lower rates by similar techniques using matrix quantization[46,47,48]. Model coders can be combined with waveform VQ to form a variety of hybrid and adaptive systems, e.g., [49,50,41,51,17].

VARIABLE RATE VQ

In many coding applications the activity of the data can vary widely over time. In such applications it is often useful to use variable rate codes, that is, codes that can use more bits for active vectors and fewer bits for dormant ones. The cost is in added buffering and software to ensure synchronization and possibly to meet a fixed rate communication channel requirement, but this cost is often justified by significant performance improvement. One approach is to simply combine VQ with traditional noiseless coding techniques. (See, e.g., [52].) Another approach is to design an inherently variable-rate VQ by using tree pruning algorithms from the theory of decision tree design to produce optimal variable-length subtrees from tree-

structured vector quantizers [53]. Here one which first designs a fixed rate tree-structured VQ using a Lloyd algorithm and then "prunes" the resulting tree using an extension of a decision tree design technique of Breiman, Friedman, Olshen, and Stone [23]. By removing the leaves of a complete tree in an optimal fashion, one obtains a collection of codes with distortion-rate pairs that can strictly dominate even full searched unstructured VQ. The resulting system is simple in comparison with noiseless codes for such large alphabets and has the successive approximation property of TSVQ that each bit improves the fidelity. Hence such codes are well suited to applications where one wishes to improve the quality of an image (or a selected portion) as additional bits arrive. Traditionally such successive improvement has been accomplished by transform coding sending additional coefficients. The VQ approach has the potential for a smoother and locally optimal sequence of improvements.

WHERE NEXT?

The emphasis in VQ appears to have shifted from research to development, in particular to real-time VLSI implementations of speech and image coders at low and medium rates. In spite of this drift, several interesting possible directions for future research exist, among them being:

1. **Fine tuning and comparing the many VQ variations with traditional transform techniques for a variety of data types, e.g., SAR, medical, video, and multispectral images.**

2. Combined VQ and signal processing (e.g., transforming and windowing and the incorporation of models for human vision and voice into the signal processing and distortion measures) for the best possible quality compression at target bit rates.

3. Real time implementation using state-of-the art custom and

semi-custom chips.

4. Applications of VQ to speech and image recognition. Understanding the theoretical connections between clustering with minimum discrimination information distortion measures and Markov source modeling. Designing speech and image compression systems that are matched to subsequent processing by machine or human experts.

5. Using digital or analog associative or Hopfield memories for VQ implementation. Since VQ does not suffer much if a code word close to the nearest neighbor is selected instead of an exact nearest neighbor (unlike the case in error control coding), associative memories are well suited to this application.

ACKNOWLEDGEMENT

Much of the research described here was supported by the National Science Foundation and by ESL, Inc.

REFERENCES

1) Shannon, C.E., "A Mathematical Theory of Communication," Bell Systems Technical Journal, vol. 27, pp. 379-423,623-656, 1948.

2) Gallager, R.G., Information Theory and Reliable communication, John Wiley and Sons, New York 1968.

3) Berger, T., Rate Distortion Theory, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1971.

4) Gray, R.M., "Vector Quantization," IEEE ASSP Magazine, Vol. 1, pp. 4-29, April 1984.

5) Gersho, A. and V. Cuperman, "Vector Quantization: A pattern matching technique for speech coding," IEEE Communications Magazine, vol. 21, pp. 15-21, December 1983.

6) Makhoul, J., S. Roucos, and H. Gish, "Vector Quantization in Speech Coding," Proceedings of the IEEE, vol. 73. No. 11, pp. 1551-1587, November 1985.

7) Gersho, A. and R. M. Gray, Signal Coding, Quantization, and Compression, 1987. In Preparation.

8) Lukaszewicz, J. and H. Steinhaus, "On Measuring by Comparison," Zastos. Mat., vol. 2, pp. 225-231, 1955. (In Polish.)

9) Lloyd, S.P., Least squares quantization in PCM. Unpublished Bell Laboratories Technical Note, 1957. Portions presented at the Institute of Mathematical Statistics Meeting Atlantic City New Jersey September 1957. Published in the March 1982 special issue on quantization of the IEEE Transactions on

Information Theory.

10) Gray, R.M., J. C. Kieffer, and Y. Linde, "Locally Optimal Block Quantizer Design," Information and Control, vol. 45, pp. 178-198, May 1980.

11) Sabin, M.J. and R. M. Gray, "Global Convergence and Empirical Consistency of the Generalized Lloyd Algorithm," IEEE Transactions on Information Theory, vol. IT-32, pp. 148-155, March 1986.

12) MacQueen, J., "Some Methods for Classification and Analysis of Multivariate Observations," Proc. of the Fifth Berkeley Symposium on Math. Stat. and Prob., vol. 1, pp. 281-296, 1967.

13) Chen, D.T.S., "On Two or More Dimensional Optimum Quantizers," Proceedings, 1977 International Conference on Acoustics, Speech, and Signal Processing, pp. 640-643, Hartford, CT, 1977.

14) Linde, Y., A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," IEEE Transactions on Communications, vol. COM-28, pp. 84-95, January 1980.

15) Tou, J.T. and R. C. Gonzales, Pattern Recognition Principles, Addison-Wesley, Reading, MA, 1974.

16) Chang, P.G. and R. M. Gray, "Gradient Algorithms for Designing Predictive Vector Quantizers," IEEE Transactions on Acoustics Speech and Signal Processing, vol. ASSP-34, pp. 679-690, August 1986.

17) Davidson, G., M. Yong, and A. Gersho, "Real Time Vector Excitation Coding of Speech at 4800 bps," Proc. ICASSP, April 1987.

18) Davidson, G., P. Cappello, and A. Gersho, Systolic Architectures for Vector Quantization. Submitted for possible publication.

19) Gersho, A. and D. Cheng, "Fast Nearest Neighbor Search for Nonstructured Euclidean Codes," Abstracts of the 1983 IEEE International Symposium on Information Theory, p. 88, St. Jovite, Quebec, Canada, September 1983.

20) Buzo, A., A. H. Gray Jr. and R. M. Gray and J. D. Markel, "Speech Coding Based Upon Vector Quantization," IEEE Transactions on Acoustics Speech and Signal Processing, vol. ASSP-28, pp. 562-574, October 1980.

21) Gray, R.M. and H. Abut, "Full Search and Tree Searched Vector Quantization of Speech Waveforms," Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing, pp. 593-596, Paris, May 1982.

22) Adoul, J.P., J. L. Debray and D. Dalle, "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, Denver Colorado, April 1980.

23) Breiman, L., J. H. Friedman, R. A. Olshen and C. J. Stone, Classification and Regression Trees, Wadsworth, Belmont, California, 1984.

24) Juang, B.H. and A. H. Gray Jr., "Multiple Stage Vector Quantization for Speech Coding," Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing, vol. 1, pp. 597-600, Paris, April 1982.

25) Gersho, A. and Y. Shoham, "Hierarchical Vector Quantization of Speech with Dynamic Codebook Allocation," Proceedings 1984 ICASSP, San Diego, March 1984.

26) Chang, P.C., J. May and R. M. Gray, "Hierarchical Vector Quantizers with Table-lookup Encoders," Proceedings 1985 IEEE International Conference on Communications, vol. 3, pp. 1452-1455, June 1985.

27) Baker, R.L. and R. M. Gray, "Differential Vector Quantization of Achromatic Imagery," Proceedings of the International Picture Coding Symposium, March 1983.

28) Sabin, M.J. and R. M. Gray, "Product Code Vector Quantizers for Waveform and Voice Coding," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-32, pp. 474-488, June 1984.

29) Chang, P.C., R. M. Gray and J. May, "Fourier Transform Vector Quantization for Speech Coding," IEEE Transactions on Communications, pp.1059-1068, 1987.

30) Heron, C.D., R. E. Crochiere and R. V. Cox, "A 32-Band Subband/Transform Coder Incorporating Vector Quantization for Dynamic Bit Allocation," Proceedings ICASSP, pp. 1276-1279, Boston, April 1983.

31) Kim, C.S., J. Bruder, M. J. T. Smith and R. M. Mersereau, "Subband Coding of Color Images using Finite State Vector Quantization," Proceedings ICASSP, pp. 753-756, 1988.

32) Budge, S.E., T. J. Stockham, Jr., D. M. Chabries and R. W. Christiansen, "Vector Quantization of Color Digital Images within a Human Visual Model," Proceedings ICASSP, pp. 816-819, 1988.

33) Cuperman, V. and A. Gersho, "Vector Predictive Coding of Speech at 16 Kb/s," IEEE Transactions on Communications, vol. COM-33, pp. 685-696, July 1985.

34) J. Foster, R. M. Gray, and M. O. Dunham, "Finite-state Vector Quantization for Waveform Coding," IEEE Transactions on Information Theory, vol. IT-31, pp. 348-359, May 1985.

35) Haoui, A. and D. G. Messerschmitt, "Predictive Vector Quantization," Proceedings ICASSP, vol. 1, pp. 10. 10. 1-10. 10. 4, San Diego, March 1984.

36) Ramamurthi, B. and A. Gersho, "Classified Vector Quantization of Images," IEEE Transactions on Communications, vol. COM-34, pp. 1105-1115, November 1986.

37) Aravind, R. and A. Gersho, "Image Compression Based on Vector Quantization with Finite Memory," Optical Engineering, vol. 26, pp. 570-580, July 1987.

38) Kim, Taejeong, "New Finite State Vector Quantizers for Images," Proceedings ICASSP, pp. 1180-1183, 1988.

39) Shen, H.-H. and R. L. Baker, "A Finite State/frame Difference Interpolative Vector Quantizer for Low Rate Image Sequence Coding," Proceedings ICASSP, pp. 1188-1191, 1988.

40) Gabor, G. and Z. Gyorfi, Recursive Source Coding, Springer-Verlag, New York, 1986.

41) Stewart, L. C., R. M. Gray and Y. Linde, "The Design of Trellis Waveform Coders," IEEE Transactions on Communications, vol. COM-30, pp. 702-710, April 1982.

42) Bei, C.D. and R. M. Gray, "Simulation of Vector Trellis Encoding Systems," IEEE Trans. on Communications, vol. COM-34, pp. 214-218, March 1986.

43) Ayanoglu, E. and R. M. Gray, "The Design of Predictive Trellis Waveform Coders Using the Generalized Lloyd Algorithm," IEEE Transactions on Communications, vol. COM-34, pp. 1073-1080, November 1986.

44) Freeman, G. H., "The Design of Time-invariant Trellis Source Codes," Abstracts of the 1983 IEEE International Symposium on Information Theory, pp. 42-43, St. Jovite, Quebec, Canada, September 1983.

45) Freeman, G. H., "Design and Analysis of Trellis Source Codes," Ph. D. Dissertation, University of Waterloo, Waterloo, Ontario, Canada, 1984.

46) Wong, D. Y., B. H. Juang and D. Y. Cheng, "Very Low Data Rate Speech Compression with LPC Vector and Matrix Quantization," Proceedings ICASSP, pp. 65-68, April 1983.

47) Tsao, C. and R. M. Gray, "Matrix Quantizer Design for LPC Speech Using the Generalized Lloyd Algorithm," IEEE Transactions on Acoustics Speech and Signal Processing, vol. ASSP-33, No. 3, pp. 537-545, June 1985.

48) Tsao, C. and R. M. Gray, "Shape-Gain Matrix Quantizers for LPC Speech," IEEE Transactions on Acoustics Speech and Signal Processing, vol. ASSP-34, No. 6, pp. 1427-1439, December 1986.

49) Rebolledo, G., R. M. G, "A Multirate Voice Digitizer Based Upon Vector Quantization," IEEE Transactions on Communications, vol. COM-30, pp. 721-727, April 1982.

50) Stewart, L. C., "Trellis Data Compression," Stanford Information Systems Lab Technical Report L905-1, July 1981.

51) Shroeder, M. R. and B. S. Atal, "Code-Excited Linear Prediction (CELP)," Proceedings ICASSP, Tampa, 198 pp. 1156-1159, 1988.

52) Ho, Y.S. and A. Gersho, "Variable-Rate Vector Quantization for Image Coding," Proceedings 1988 ICASSP, pp. 1156-1159, 1988.

53) Chou, P. A., T. Lookabaugh and R. M. Gray, "Optimal Pruning with Applications to Tree-structured Source Coding and Modeling," IEEE Transactions on Information Theory, to appear.

54) Gray, R. M., "Tree-searched Block Source Codes," Proceedings of the 1980 Allerton Conference, Allerton, Il., Oct. 1980.

55) Blahut, R. E., Principles and Practice of Information Theory, Addison-Wesley, Reading, Mass. 1987.

56) Witten, J. H., R. M. Neal and J. G. Cleary, "Arithmetic Coding for Data Compression," Communications of the ACM, pp. 520-540, Vol. 30.

57) Welch, T. A., "A Technique for High-performance Data Compression," Computer, pp.8-18, 1984.

58) Ziv, J. and A. Lempel, "Compression of Individual Sequences via Variable-rate Coding," IEEE Transactions on Information Theory, Vol. IT-24, 1978.

59) Rice, R. F. and J. R. Plaunt, "Adaptive Variable Length Coding for Efficient Compression of Spacecraft Television Data," IEEE Transactions on Commun. Tech., Vol. COM-19, pp. 889-897, 1971.

60) Rice, R. F., "Practical Universal Noiseless Coding," SPIE Symposium Proceedings, Vol. 207, San Diego, CA, August 1979.

61) Storer, J., Data Compression, Computer Science Press, 1988.

62) Shannon, C. E., Coding theorems for a discrete source with a fidelity criterion," IRE Natl. Conv. Record, Part 4, pp. 142-163, 1959.

63) Equitz, W., "Fast Algorithms for Vector Quantization Picture Coding," Proceedings ICASSP, pp. 18.1.1-18.1.4, April 1987.

64) Equitz, W., "Fast Algorithms for Vector Quantization Picture Coding," IEEE Transactions on ASSP, in review.