AT&T Technical Journal
Vol. 64, No. 5, May–June 1985
Printed in U.S.A.

On the Use of Vector Quantization for Connected-Digit Recognition

By S. C. GLINSKI*

(Manuscript received June 6, 1984)

Recent work at AT&T Bell Laboratories has demonstrated the efficacy of vector quantization in greatly reducing both the computational and memory requirements of isolated-word recognition systems. This efficiency is obtained at the expense of a marginal decrease in performance, and thus is an attractive approach. The purpose of this paper is to report on the results of a series of experiments in the application of vector-quantization strategies to a small-vocabulary, connected-word recognition task. Several strategies are investigated, including the use of speaker-trained code books versus universal code books, the use of binary and higher-order tree searches versus full searches of these code books, and the quantization of both test and reference frames versus reference frames only. For various strategies, the effect on error rate of varying the code-book size is also reported. Results indicate that the vector quantization approach is attractive for linear predictive coding-based connected-digit recognition.

I. INTRODUCTION

In the area of speech coding, the technique of Vector Quantization (VQ) has recently been successfully applied.[1,2] In the standard approach, a speech signal is framed and a feature vector is extracted from each frame. Each element of the feature vector is then separately quantized. In other words, the value of each element is replaced by its closest match from a set of discrete values. The set of values is chosen to minimize some error criterion while reducing the number of bits

* AT&T Bell Laboratories.

Copyright © 1985 AT&T. Photo reproduction for noncommercial use is permitted without payment of royalty provided that each reproduction is done without alteration and that the Journal reference and copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free by computer-based and other information-service systems without further permission. Permission to reproduce or republish any other portion of this paper must be obtained from the Editor.

required to identify the element value. This feature vector is typically a set of Linear Predictive Coding (LPC) coefficients and perhaps an energy term. In the VQ approach, a feature vector is extracted as before. The entire vector of features is then quantized by replacing the vector with its closest match from a set or "code book" of feature vectors. Similarly, the entries in the code book are chosen to minimize some distortion measure while reducing the number of bits required to identify each frame of speech. In addition to reducing the number of bits required to represent each feature vector, by using a suitably compact code book in place of a large set of reference vectors, it is possible to greatly reduce the number of comparisons made between the unknown (test) feature vector and the stored-feature vectors. Since this comparison (or distortion measure) is the current bottleneck in many speech recognition systems, the VQ approach is quite effectively used in them. In fact, VQ has been used heavily in several different approaches to speech recognition, including Hidden Markov Modeling (HMM)[3,4] and Dynamic Time Warping (DTW),[3] among others.[5]

The purpose of this paper is to report on the results of a series of experiments in the application of VQ strategies to a small-vocabulary, connected-word recognition task.[6-7] These strategies include the use of Speaker-Dependent (SD) versus Speaker-Independent (SI) code books; the use of binary and higher-order tree searches versus full searches of these code books, as suggested in Ref. 2; and the quantization of both test and reference frames versus reference frames only.[8,9] The effect on error rate of varying the code-book size is also reported. In all experiments, the reference and test data sets are disjoint, and code books are trained with reference data. That is, speaker-dependent reference templates are quantized by vocabulary-dependent code books that are either speaker dependent or speaker independent. All test strings consist of deliberately spoken, connected words.

In Section II some useful terminology is presented. In Section III theory is developed. In Section IV experimental results are presented.

II. TERMINOLOGY

This terminology is based, in part, on Refs. 7 and 10.

p	LPC order
M^*	maximum code-book size in words
M	code-book size in words $2 \leq M \leq M^*$
k	$\text{Log}_2 M$ code-book rate
i	code-word (reference) index $0 \leq i \leq M-1$
$I(k)$	best code-word match at code book k
j	training-frame index $1 \leq j \leq J$
J	number of training frames

B	branching factor
α	$\text{Log}_2 B$
$a_i^k(l)$	LPC reference vector $0 \le l \le p$
$r_i^k(l)$	LPC reflection coefficient vector $0 \le l \le p$
V_j	autocorrelation matrix of training frame
E_j	LPC residual energy of training frame
$R_j(l)$	autocorrelation vector of training frame
$d(a_i, V_j)$	distortion between reference and training frames
$\rho(l)$	perturbation vector
$\bar{D}_J$	mean distortion over training frames
ϵ	distortion change threshold (0.01)
δ	perturbation factor (0.01)
$\{T_M(i)\}$	set of training vectors whose best match is code word i
$C_M(i)$	number of training vectors in $\{T_M(i)\}$.

III. VECTOR QUANTIZATION

The VQ procedure consists of two main parts: code-book generation and the classification of test frames. In both cases, a code-book search procedure must be employed to classify the training or test (unknown) frames, respectively. The code-book generation process will be presented first and the search strategy second.

3.1 Code-book generation

The basic generation procedure discussed in Refs. 1, 2, and 10 is employed and is illustrated in Fig. 1. The flowchart is from Ref. 10, but has been generalized to allow B-way splitting of centroids versus the original two-way splitting (B is a power of 2 in this work). The overall goal is to find a set of code words $\{a\}$, such that the mean distortion $\bar{D}_J$, produced by replacing each of the J training frames by its closest match from the code book $\{a\}$, is minimized. Succinctly stated,

$$\bar{D}_J(M^*) = \min_{\{a\}} \left[\frac{1}{J} \sum_{j=1}^{J} \min_{1 \le i \le M^*} d(a_i, V_j) \right]. \tag{1}$$

The distortion d can be calculated using the likelihood-ratio-distance metric as follows:

$$d(a_i, V_j) = \frac{a_i V_j a_i^t}{a_j V_j a_j^t} - 1, \tag{2}$$

where row m, column n of matrix V_j contains $(1/N)R_j(|m-n|)$. N is the window length and R is the autocorrelation vector. Only the spectral-shape information is used in the quantizer.

In practice, $\{a\}$ is found by first initializing with a small code book

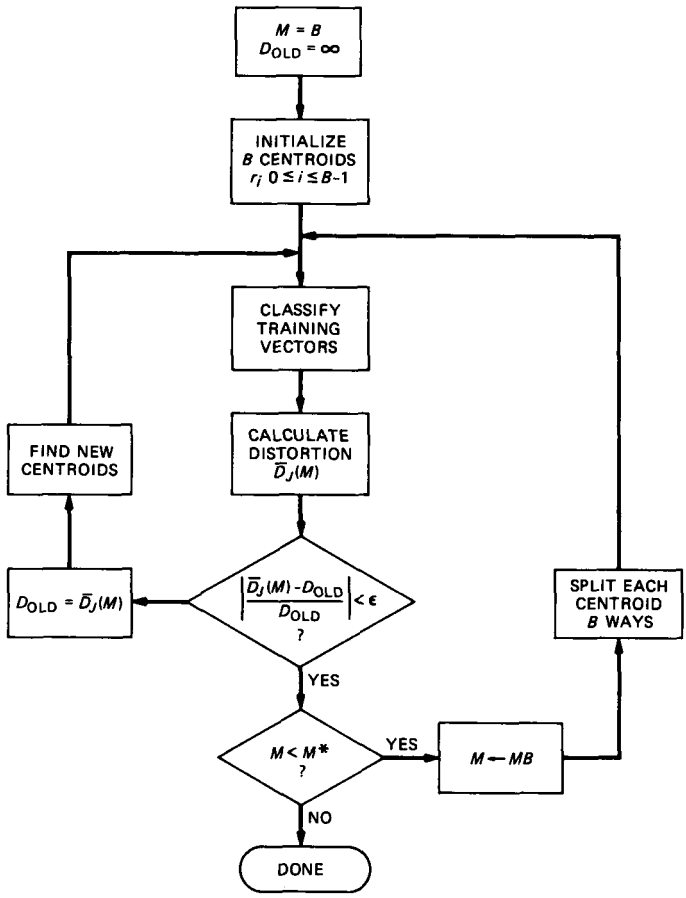


Fig. 1—Code-book generator flowchart.

of size $M = B$ (where B is typically 2), and then successively splitting its code words B ways to obtain larger and larger code books until a code book of the desired size is obtained (see outer loop of Fig. 1). Each successive code book is corrected by iteratively classifying the training set, and adjusting each code word to be the centroid of the subset of training frames which best match that code word (inner loop). The final iteration is determined as that for which the mean distortion changes by less than a threshold ϵ in relation to the previous mean distortion. Typically, $\epsilon = 0.01$.

Initialization of centroids is as follows:

$$\begin{aligned} r_i^k(l) &= 0.5(-1)^{\lfloor 2^{-l} i \rfloor} \qquad 0 \le l \le \mathrm{Log}_2 B - 1 \\ &= 0.0 \qquad\qquad\qquad \mathrm{Log}_2 B \le l \le p, \end{aligned} \tag{3}$$

where $k = \text{Log}_2 B$ and $0 \leq i \leq B - 1$. This splits the LPC reflection coefficient space on the first $\text{Log}_2 B$ coordinate axes. For instance, for $B = 4$,

$$r_0^2 = (0.5, 0.5, 0, \cdots 0)$$

$$r_1^2 = (-0.5, 0.5, 0, \cdots 0)$$

$$r_2^2 = (0.5, -0.5, 0, \cdots 0)$$

$$r_3^2 = (-0.5, -0.5, 0, \cdots 0).$$

Centroid splitting is done similarly as follows:

$$\begin{aligned} \rho_i(l) &= \delta(-1)^{\lfloor 2^{-l} i \rfloor} && 0 \leq l \leq \text{Log}_2 B - 1 \\ &= 0.0 && \text{Log}_2 B \leq l \leq p \\ & && 0 \leq i \leq B - 1. \end{aligned} \tag{4}$$

Then

$$\begin{aligned} r_{Bn+i}^{k+\alpha} &= r_n^k \cdot (1 + \rho_i) && 0 \leq i \leq B - 1 \\ & && 0 \leq n \leq 2^k - 1 \end{aligned} \tag{5}$$

accomplishes a B-way split of each reflection coefficient vector in code book k. The factor δ is typically set to 0.01. LPC model stability is ensured by requiring that $-1 \leq r \leq 1$.

The new centroid (code-word) computation depicted in Fig. 1 is done by averaging the normalized autocorrelations of those training frames that best match a given code word:

$$R_i(l) = (C_M(i))^{-1} \sum_j E_j^{-1} R_j(l) \quad j \in \{T_M(i)\}. \tag{6}$$

This computes the centroid as a spectral average of training frames.[2] The total-average-distortion calculation is

$$\bar{D}_J(M) = M^{-1} \sum_{i=1}^{M} (C_M(i))^{-1} \sum_j d(a_i, V_j) \quad j \in \{T_M(i)\}, \tag{7}$$

where the LPC coefficients a_i of eq. (7) are derived from the R_i of eq. (6).

The only program step in Fig. 1 not yet discussed is the classification of training vectors. Since this is common to both the code-book training and ultimate quantization of test vectors, it will be discussed in the next section.

3.2 Classification of frames

In the original development of VQ, a full search of the code book was used to classify training (or test) frames. The best matching code word in a k-bit code book is

$$I(k) = \underset{i}{\text{argmin}}\,[d(a_i^k, V)] \qquad 0 \le i \le M^* - 1. \tag{8}$$

For classification of the test, $k = \text{Log}_2 M^*$. During the code-book generation, k varies from $\alpha = \text{Log}_2 B$ to $\alpha = \text{Log}_2 M^*$, depending on how many times the code words have been split.

However, Fig. 2a shows that a binary tree search is also possible. In any given code book (level), only two code words are searched; those that were split from the best matching code word in the previous code book (level). The search is initialized by setting $I(O) = 0$, and then executing

$$I(k) = \underset{i}{\text{argmin}}\,[d(a_i^k, V)] \qquad 1 \le k \le \text{Log}_2 M^*$$
$$2[I(k-1)] \le i \le 2[I(k-1)] + 1. \tag{9}$$

The search stops at code book $k = \text{Log}_2 M^*$ for test classification, or at an intermediate k during code-book training.

The tree search may be carried out using a different branching factor, B (which in this work is specified to be an integer power of 2). Initialize $I(O) = 0$, and then

$$I(k) = \underset{i}{\text{argmin}}\,[d(a_i^k, V)] \qquad k = \alpha, 2\alpha, \cdots \text{Log}_2 M^*$$
$$B[I(k-\alpha)] \le i \le B[I(k-\alpha) + 1] - 1. \tag{10}$$

The stopping criteria are the same as for eq. (9). If $\text{Log}_2 M^* \neq \alpha\beta$ for some positive integer β, then the branching factor to the last code book (level) is not $B = 2^\alpha$, but $B = 2^\gamma$, $\gamma < \alpha$, where

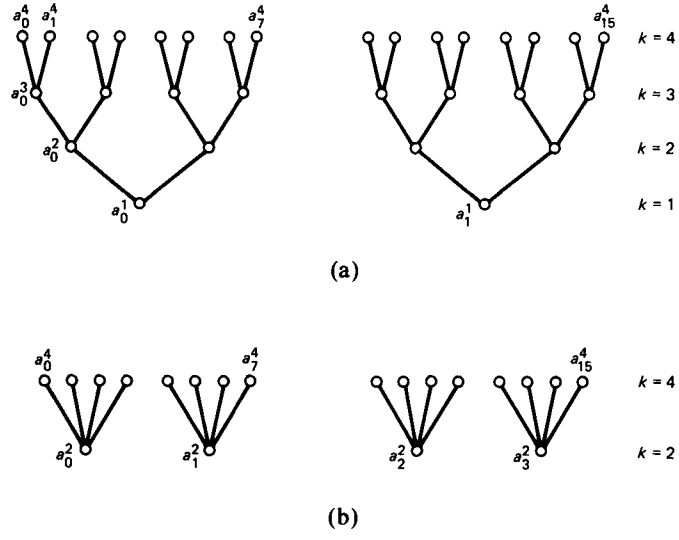
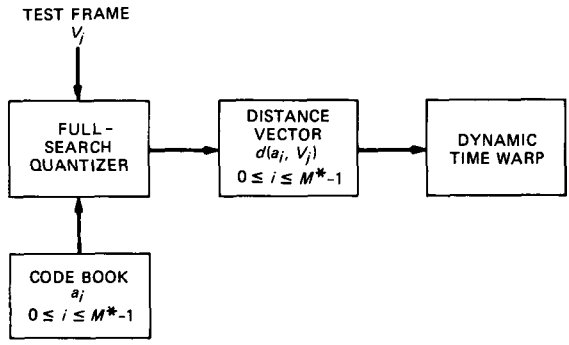


Fig. 2—Code-book tree search: (a) quaternary search; (b) binary search.

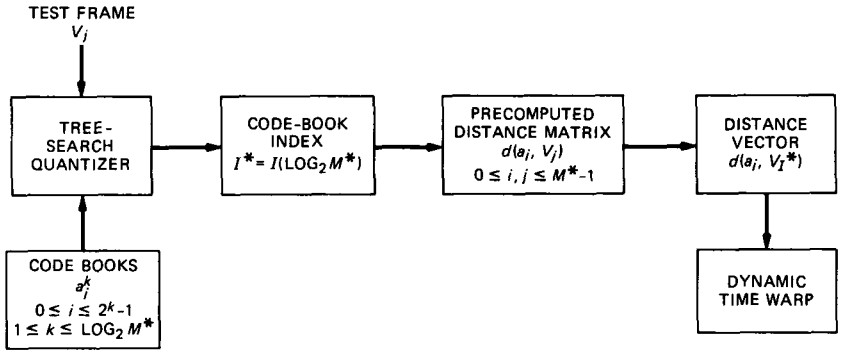
$$\gamma = \left\{ \text{Log}_2 M^* - \left\lfloor \frac{\text{Log}_2 M^*}{\alpha} \right\rfloor \alpha \right\}. \tag{11}$$

Figure 2b shows an example of the tree-search classification procedure for a branching factor of 4 ($B = 4$).

Note that a full search of a k-bit code book requires $M = 2^k$ distortion computations, while a tree search of the same requires only $B \text{ Log}_B M$ distortion computations with a branching factor of B. Thus, a great computational savings is incurred by using the tree-search strategy. The two search strategies suggest two approaches to connected-word recognition via DTW, as pictured in Fig. 3. In the following discussion an LPC distortion will be referred to as a distance, which is more common in the jargon of speech recognition. In Fig. 3a (full-search strategy) a vector of distances representing the distances between the current test frame and the entire code book is passed to the DTW procedure.[8] The vector is sufficient to represent the distance between any test and reference template frames. Since a full search was



(a)



(b)

Fig. 3—Quantizer strategies: (a) full search; (b) tree search.

employed, all these distances are available along with $I(k)$, the best matching code word. In the tree-search strategy (see Fig. 3b) this is not the case. Only the index $I(k)$ and corresponding distance $d(a_{I(k)}, V_j)$ are produced. Thus, it is necessary to precompute and store a cross table of distances between all pairs of code words. This implies that both reference and test frames are quantized to the code-book entries, unlike in Fig. 3a, where solely the reference frames are quantized. Thus, the tree-search strategy will require much less computing power but probably do more poorly than the full-search strategy, because of both the implicit quantization of the test signal and errors introduced by the tree search.

In the HMM approach, however, for the procedure analogous to time warping (Viterbi scoring, for example), the only information necessary to pass from the quantizer is the index $I(k)$ (no distance vector). Thus, there is no need for the precomputed distance matrix in the tree-search strategy, and the only error introduced results from the tree search.

3.3 Empty cluster problem

In practice, during the classification of training frames, it may happen that certain code words may not match any training frame. In other words, the set $\{T_M(i)\}$ may be empty for one or more i, and the count $C_M(i) = 0$ for the same i. This is especially a problem as the code-book size approaches the number of training frames. In this work, the problem is addressed by simply ignoring the empty clusters (which reduces the effective code-book size). During a full-search strategy, the empty clusters are simply skipped. In a tree-search strategy, however, it may happen that as the tree is traversed, a "dead end" is reached before the last level is reached. That is, all branches from a given node lead to code words corresponding to empty clusters. In this case, the test frame is classified as the code word corresponding to the last nonempty cluster encountered as the tree was traversed.

The advantage of this approach is that experiments may be run where the number of training frames is fewer than the specified number of code words. This is especially true for the speaker-dependent code-book experiments in this work.

IV. EXPERIMENTAL RESULTS

To evaluate the various VQ strategies, the connected-word recognizer of Ref. 7 was employed. It can be shown that this recognition algorithm performs identically to the level-building algorithm of Myers and Rabiner[11] in the case where no constraint is imposed on the number of words in the test. When this constraint is removed, the level-building algorithm performance is marginally superior.[12] The

distance measure used in the recognizer was the likelihood ratio of eq. (2), clipped at 2.0. The test data consist of 40 random strings, spoken by each of 18 different speakers, 9 male and 9 female. These are equal numbers of strings of length 2, 3, 4, and 5 digits. The strings used in this work are all deliberately spoken (about two words per second).

The reference data consist of 33 reference templates (about 960 reference frames) for each speaker. Three groups of 11 templates each contain the digits 0 through 9, and one repetition of the digit with an unreleased final consonant "t." The first group includes standard isolated-word reference templates; while the second contains embedded, noncoarticulated, deliberately spoken references; and the third contains embedded, coarticulated, normally spoken references. Embedded templates are those extracted from between two other templates in running speech. The three groups are used to compensate for differences in speaking rates and coarticulation effects.

Both the reference and test data were passed through an endpoint detector. All the speech data are identical to those used in previous work.[13] Note that all experiments entail speaker-dependent, connected-word recognition, and code books are trained with the reference template data. Reference templates and test data sets are disjoint.

For each of the 18 speakers, a speaker-independent code book was generated from the reference templates of the other 17 speakers. These code books are "vocabulary dependent," since the same vocabulary was included in the code-book training data as in reference and test data.

For speaker-dependent code books, the reference database for an individual speaker was used to train the code book for that speaker. This corresponds to an implementation in which reference templates for a given speaker are created first, these templates are then used to train a code book, and then the reference templates are quantized with the same code book. Thus, in this case the code books are both speaker dependent and vocabulary dependent.

Parameters of the code-book generator were chosen as follows:

$\epsilon = 0.01$ (distortion change threshold)

$\delta = 0.01$ (perturbation factor).

The baseline string error rate for the recognizer is 5.4 percent. This baseline error rate is for the case where no vector quantization is used.

V. SPEAKER-INDEPENDENT CODE BOOKS

In the first experiment, speaker-independent code books were used to quantize reference frames only (one-sided quantization[9]). Code books of rate 4, 6, 8, and 10 bits were investigated for both binary and

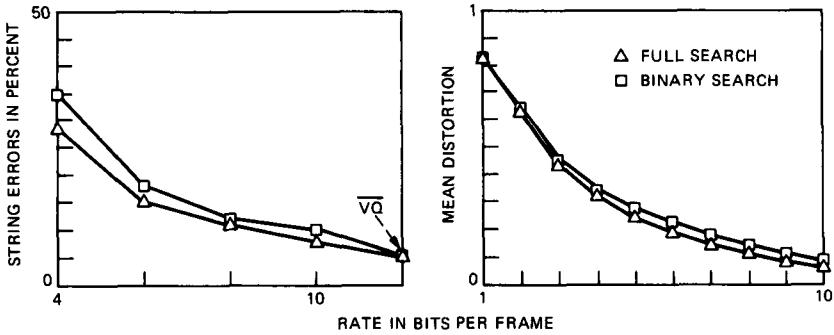


Fig. 4—Recognizer performance and code-book distortion for one-sided VQ and SI code books.

full-search classification during code-book generation and reference quantization. The mean distortion of the binary and full-search code books is shown in Fig. 4. This is the mean distance $\bar{D}_j$ between the training vectors and their best matching code words. The recognizer performance, as a function of code-book rate, is shown in Fig. 4 and listed in Table I. Results show that: (1) Only the use of a 10-bit, full-search code book results in performance approaching the baseline performance. Other rates result in approximately a doubling (or worse) in error rate. (2) About a 1-bit savings in code-book rate is obtained by using a full-search strategy versus a tree-search strategy, while recognizer performance is held constant.

VI. SPEAKER-DEPENDENT CODE BOOKS

The second experiment was identical to the first except that speaker-dependent code books were used in place of speaker-independent code books. The mean distortion of the binary- and full-search code books is shown in Fig. 5. The recognizer performance as a function of code-book rate is shown in Fig. 5, and listed in Table I. Results indicate the following: (1) The recognizer shows little loss in performance at rates of 8 bits and higher. This represents a compression on the reference

Table I—Recognizer performance

Code Book	VQ	Search	Percent String Errors Rate (Bits/Frame) 4	6	8	10
SI	1 Side	Full	28.3	15.1	11.0	8.1
SI	1 Side	Binary	34.9	18.2	12.1	10.1
SD	1 Side	Full	15.3	9.3	6.0	5.7
SD	1 Side	Binary	22.1	9.6	6.4	5.6
SD	2 Side	Full	21.0	10.8	7.5	5.8
SD	2 Side	Binary	31.2	14.7	10.7	9.3

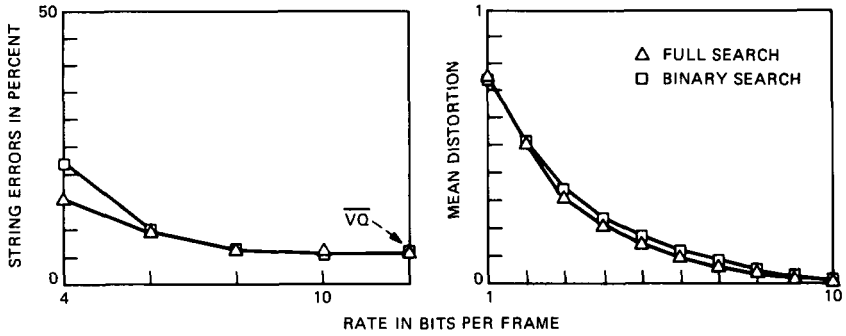


Fig. 5—Recognizer performance and code-book distortion for one-sided VQ and SD code books.

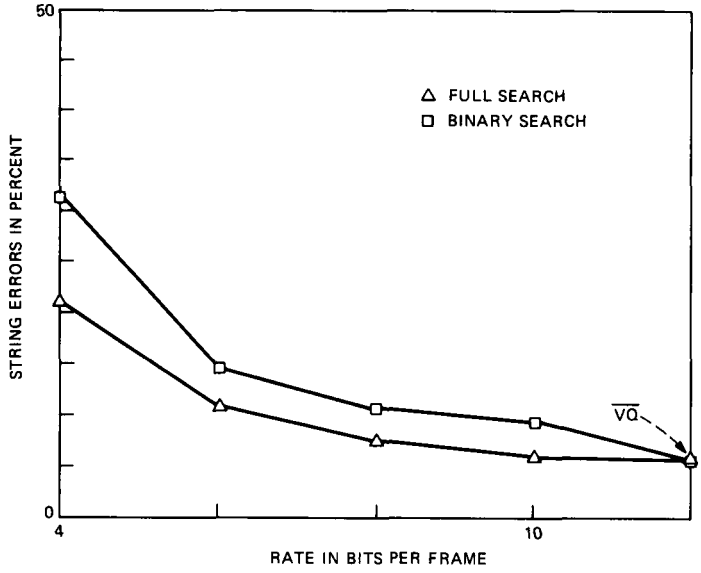


Fig. 6—Recognizer performance for two-sided VQ and SD code books.

database of about 4:1 for each speaker. (2) The binary tree-search strategy rivals the full-search strategy with regard to recognizer performance at rates of 6 bits and up. (3) For a given performance, the SD code book incurred a bit-rate savings from 1 bit (at a low-bit rate) to 3 bits (at a high-bit rate) over the SI code book.

VII. REFERENCE AND TEST QUANTIZATION

The third experiment was identical to the second, except that both reference and test frames were quantized (two-sided quantization[9]). Recognizer performance is plotted in Fig. 6 and listed in Table I. The

Table II—Recognizer performance using tree-search strategies

Branching Factor					
Full search	2	4	8	16	
6.0	6.4	6.5	6.9	7.4	% String errors

results show the following: (1) Two-sided quantization in the full-search case rivals one sided for bit rates of 6 and up. (2) The binary-search case does not compete as favorably with the full-search approach as in experiment 2.

VIII. TREE-SEARCH EXPERIMENT

In experiment 4, experiment 2 was repeated for the 8-bit code-book rate, while the branching factor B in the tree-search approach was varied between 2 and 16. Only reference frames were quantized (SD code book). Performance data listed in Table II and plotted in Fig. 7 indicate that performance drops off only slightly with an increasing branching factor.

IX. CONCLUSIONS

Four experiments were run to determine the efficacy of VQ as applied to a small-vocabulary, connected-word recognition task. Re-

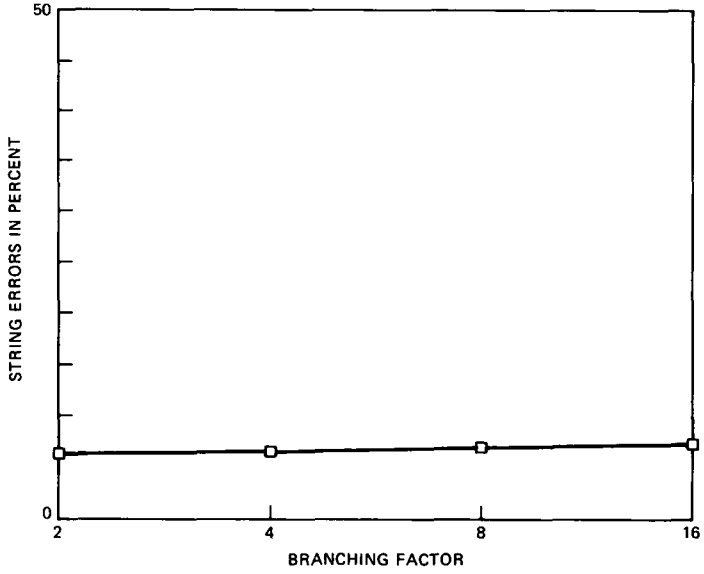


Fig. 7—Recognizer performance for tree-search strategies using one-sided VQ and SD code books at 8 bits per frame.

sults show that both full- and binary-search, one-sided VQ, at codebook rates of 6 bits and higher, offers an attractive cost-performance trade-off in an LPC-based connected-digit recognizer. Full-search two-sided VQ is also a competitive approach. For tree-search strategies, recognizer error rate is relatively insensitive to branching factor.

REFERENCES

1. Y. Linde, A. Buzo, and R. Gray, "An Algorithm for Vector Quantizer Design," IEEE Trans. Commun., *COM-28,* No. 1 (January 1980), pp. 84–5.
2. A. Buzo et al., "Speech Coding Based Upon Vector Quantization," IEEE Trans. Acoust., Speech, and Signal Processing, *ASSP-28,* No. 5 (October 1980), pp. 562–74.
3. L. R. Rabiner, S. E. Levinson, and M. M. Sondhi, "On the Application of Vector Quantization and Hidden Markov Models to Speaker-Independent, Isolated Word Recognition," B.S.T.J., *62,* No. 4 (April 1983), pp. 1075–105.
4. S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process in Automatic Speech Recognition," B.S.T.J., *62,* No. 4 (April 1983), pp. 1035–74.
5. J. Shore and D. Burton, "Discrete Utterance Speech Recognition Without Time Alignment," IEEE Trans. Information Theory, *IT-29,* No. 4 (July 1983), pp. 473–91.
6. T. K. Vintsyuk, "Element-wise Recognition of Continuous Speech Consisting of Words of a Given Vocabulary," Kibernetika (Cybernetics), 7, No. 2 (1971), pp. 361–72.
7. J. S. Bridle, M. D. Brown, and R. M. Chamberlain, "An Algorithm for Connected Word Recognition," Proc. 1982 ICASSP, pp. 899–902.
8. H. Sakoe, "Device for Recognizing and Input Pattern With Approximate Patterns Used for Reference Patterns on Mapping," U.S. Patent 4,256,924, March 17, 1981.
9. K. Shikano, "Spoken Word Recognition Based Upon Vector Quantization of Input Speech," Trans. Committee on Speech Research (December 1982), pp. 473–80.
10. L. R. Rabiner, M. M. Sondhi, and S. E. Levinson, "Note on the Properties of a Vector Quantizer for LPC Coefficients," B.S.T.J., *62,* No. 8 (October 1983), pp. 2603–16.
11. C. S. Myers and L. R. Rabiner, "A Level Building Dynamic Time Warp Algorithm for Connected Word Recognition," IEEE Trans. Acoust., Speech, and Signal Processing, *ASSP-29,* No. 2 (April 1981), pp. 284–97.
12. S. Glinski, "A Comparison of Dynamic Time Warp Algorithms for Connected Word Recognition," paper presented at 106th meeting of Acoustical Society of America, San Diego, Calif., November 1983.
13. L. R. Rabiner, A. Bergh, and J. G. Wilpon, "An Embedded Word Training Procedure for Connected Digit Recognition," Proc. 1982 ICASSP, pp. 1621–4.

AUTHOR

Stephen C. Glinski, B.S. (summa cum laude, Electrical Engineering), 1977, Lowell Technological Institute, Mass.; M.S., 1978, Ph.D., 1981 (Electrical Engineering), University of Illinois at Urbana-Champaign; AT&T Bell Laboratories, 1982—. While at the University of Illinois, Mr. Glinski was a research assistant at the Computer Education Research Laboratory involved in the area of speech synthesis. At AT&T Bell Laboratories, he has been with the Speech Recognition Group engaged in the development of efficient algorithms and architectures for connected-word speech recognition. Member, IEEE, Eta Kappa Nu.