An Adaptive, Ordered, Graph Search Technique for Dynamic Time Warping for Isolated Word Recognition

MICHAEL K. BROWN AND LAWRENCE R. RABINER, FELLOW, IEEE

Abstract—The technique of dynamic time warping (DTW) is relied on heavily in isolated word recognition systems. The advantage of using DTW is that reliable time alignment between reference and test patterns is obtained. The disadvantage of using DTW is the heavy computational burden required to find the optimal time alignment path. Several alternative procedures have been proposed for reducing the computation of DTW algorithms. However, these alternative methods generally suffer from a loss of optimality or precision in defining points along the alignment path. In this paper we propose another alternative procedure for implementing a DTW algorithm. The procedure is based on the well-known class of techniques for a directed search through a grid to find the "shortest" path. An adaptive version of a directed search procedure is defined and shown to be capable of obtaining the exact DTW solution with reduced computation of distances but with increased overhead. It is shown that for machines where the time for distance computation is significantly larger than the time for combinatorics and overhead, a potential gain in speed of up to 3:1 can be realized with the directed search algorithm. Formal comparison of the directed search algorithm with a standard DTW method, in an isolated word recognition test, showed essentially no loss in recognition accuracy when the parameters of the directed search were selected to realize the 3:1 reduction in distance computation.

I. Introduction

IT IS well known in the area of speech recognition that optimal time alignment of reference patterns to test patterns substantially reduces recognition errors for a vocabulary with polysyllabic words [1]. Typically, time alignment is performed on speech data which is represented as a time sequence of feature vectors (e.g., vectors of linear prediction coefficients) which represent the spectral information in corresponding "frames" of the speech signal. The most commonly used time alignment procedures, for the speech recognition problem, are the class of algorithms referred to as dynamic programming (DP) or dynamic time warping (DTW) methods [2]-[5]. As shown in Fig. 1, these procedures require calculation of the local distance between each possible reference and test frame (within a prescribed global range, e.g., the parallelogram of Fig. 1), in order to determine the optimal time alignment path relating reference and test frames.

For most isolated word recognition systems using DTW for time alignment, it has been shown that the DTW algorithm requires the vast majority of the processing time required to recognize a word. Consequently, several attempts have been made to modify the DTW algorithm to eliminate some of the computation either by being more restrictive with the DTW constraints [3]-[5], or by approximating the reference pattern by *states* that are variable in duration [6]-[7]. In either case, efficiency is gained at the sacrifice of optimal time alignment. As a result, recognition error rate generally increases over that obtained using the full DTW alignment algorithm.

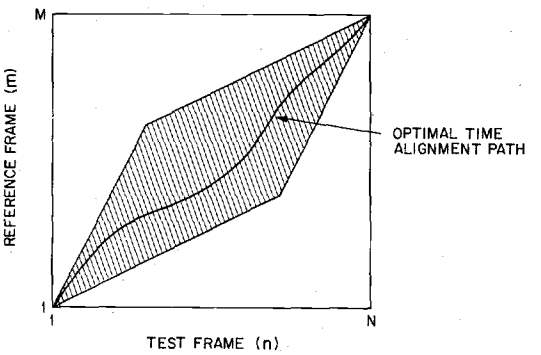


Fig. 1. Region in the (n, m) plane for which a time alignment contour is calculated in dynamic time warping.

In this paper we present a new approach to finding an optimal time alignment path which can substantially reduce computation without sacrificing optimality of the resulting path. The way in which these efficiencies are achieved is by modeling the DTW problem as one of finding a directed path through a constrained grid. By modeling the grid as a digraph with conditional branch costs (or equivalently production rules), an ordered graph searching (OGS) algorithm can be used to solve for the best path through the grid. It will be shown that such an algorithm can be designed to guarantee essentially optimal time alignment while reducing computation over that required for a conventional dynamic programming (DP) solution. Furthermore, we will show that by slightly relaxing the path optimally conditions, a substantial reduction in computation (>60 percent) can be achieved with only a small loss in accuracy.

The ordered graph search type of algorithm, of the type to be described in this paper, is most useful for implementations of a word recognizer where distance calculations made on local features of the speech pattern (e.g., LPC coefficients) are computationally expensive (e.g., simple microprocessor systems). In such cases the control overhead is relatively inexpensive compared to the cost of distance calculation. Then

Manuscript received August 3, 1981.
The authors are with Bell Laboratories, Murray Hill, NJ 07974.

0096-3518/82/0800-0535$00.75

the reduction in computation for local distances is approximately 65 percent, at the expense of doubling the control overhead, and about 4.5 kbytes of additional memory. For implementations where the cost of local distances is negligible (e.g., using a peripheral array processor or peripheral high speed multiplier), the increased cost of overhead can be comparable to the decreased cost of local distances. For such cases there would be little advantage to the proposed algorithm.

The organization of this paper is as follows. In Section II we review the "standard" DTW algorithm as this provides the basis of comparison for the OGS algorithm to be presented in Section III. In Section IV we describe the results of a series of isolated word recognition tests designed to compare speed and accuracy of the two time alignment procedures. Finally, in Section V we discuss the advantages and disadvantages of the OGS algorithm relative to the standard DP method, for various implementations.

II. The Standard DTW Algorithm

We assume that we are given a test pattern $\boldsymbol{T}$, consisting of a sequence of N vectors, i.e.,

$$\boldsymbol{T} = \{T(1), T(2), \cdots, T(N)\} \tag{1}$$

where the vector $T(i)$ is a spectral representation of the ith frame of the test word. In our system the vectors $T(i)$ are a set of nine autocorrelations (from which an eighth order LPC model is derived). Alternative $T(i)$ vectors include filter bank energies, cepstral coefficients, etc. The duration of the test word is N frames, where each frame represents 45 ms of speech, and adjacent frames are spaced 15 ms apart.

For a given vocabulary of V words, we denote the reference pattern for the vth word as $\boldsymbol{R}_v$, and we again represent each reference pattern as a sequence of M_v vectors, i.e.,

$$\boldsymbol{R}_v = \{R_v(1), R_v(2), \cdots, R_v(M_v)\} \tag{2}$$

where each vector is again a spectral representation of the corresponding frame within the word.

In order to optimally align the time scales of the test (the n index) and reference (the m index) patterns via a dynamic time warping algorithm, we must solve for a warping, or path alignment function of the form

$$m = w(n) \tag{3}$$

and thereby seek to minimize the total distance

$$D = \sum_{n=1}^{N} \tilde{d}(T(n), R(w(n))) \tag{4}$$

over all possible $w(n)$, where $\tilde{d}$ is the local distance between test frame n and reference frame $m = w(n)$. To solve the DTW problem requires specification of the following [5]:

1) endpoint constraints;
2) local path constraints;
3) global path constraints;
4) axis orientation;
5) local distance measure.

We have considered one form of DTW algorithm with the following specifications.

Endpoint Constraints: We assume that the word endpoints of both the test and reference patterns have been accurately determined, and so we require the path to obey the constraints

$$w(1) = 1 \tag{5a}$$

$$w(N) = M. \tag{5b}$$

Local Path Constraints: We assume the Itakura path constraints [2] are obeyed, namely,

$$0 \leqslant w(n) - w(n-1) \leqslant 2 \tag{6a}$$

$$w(n) - w(n-1) = 0 \quad \text{iff} \quad w(n-1) - w(n-2) > 0. \tag{6b}$$

These local path constraints guarantee that the average slope of the warping function lies between $\frac{1}{2}$ and 2, and guarantee path monotonicity.

Global Path Constraints: The endpoint conditions (5) and the local path constraints (6) lead to a set of global path constraints of the form

$$m_L(n) \leqslant m \leqslant m_H(n) \tag{7}$$

where

$$m_H(n) = \min\{2(n-1)+1, M - \tfrac{1}{2}(N-n), M\} \tag{8a}$$

$$m_L(n) = \max\{\tfrac{1}{2}(n-1)+1, M - 2(N-n), 1\}. \tag{8b}$$

These global path constraints essentially define the parallelogram of Fig. 1 with lines of slope 2 and $\frac{1}{2}$ emanating from the points $m = 1, n = 1$ and from $m = M, n = N$.

Axis Orientation: We assume that the test sequence index n is always mapped to the abscissa (the independent variable) and the reference sequence index m is always mapped to the ordinate (the dependent variable). Experience has shown this orientation to lead to the best performance in word recognition systems [4], [5].

Local Distance Measure: The local distance used in this study is the Itakura log likelihood ratio [2], which is implemented in the form

$$\tilde{d}(T(n), R(m)) = \log\,[T(n) \cdot R(m)], \tag{9}$$

i.e., a log of the dot product of the two vectors $T(n)$ and $R(m)$.

Word recognition is achieved by computing the optimal warping path and distance for each word in the vocabulary, giving

$$D_v = \min_{w(n)} \left[\sum_{n=1}^{N} \tilde{d}(T(n), R_v(w(n))) \right] \tag{10}$$

and using the nearest neighbor rule to choose word v^* as the best candidate where

$$v^* = \operatorname*{argmin}_{v} [D_v]. \tag{11}$$

An ordered set of word distances is also maintained for statistical analysis purposes.

One additional modification was made to the recognition algorithm and that was to use the normalize-and-warp procedure of Myers *et al.* [5] prior to the DTW alignment. For this procedure the test pattern, and each reference pattern, is linearly warped to a fixed duration (the average word duration

over all words in the vocabulary) prior to DTW alignment, thereby ensuring that the widest range of time alignment paths is considered. Experimental results have shown this procedure to be valid on several recognition vocabularies [5].

A. The Basic DTW Iteration

Based on the above specifications, the techniques of dynamic programming can be used to solve (10) iteratively for the best path as follows. We define an accumulated distance function $D_A(n, m)$ as the total distance from the grid point (1, 1) to the grid point (n, m), along the *best path* between these grid points (minimum distance), and we can use the iteration (based on the local constraints)

$$D_A(n, m) = \tilde{d}(T(n), R(m)) + \min [D_A(n-1, m)\, g(n-1, m), D_A(n-1, m-1), D_A(n-1, m-2)]$$
$$1 \leqslant n \leqslant N, \quad m_L(n) \leqslant m \leqslant m_H(n) \tag{12}$$

where $g(n, m)$ is the nonlinear weighting

$$g(n, m) = \begin{cases} 1 & \text{if } w(n) \neq w(n-1) \\ \infty & \text{if } w(n) = w(n-1) \end{cases} \tag{13}$$

to guarantee that the optimum path to (n, m) does not stay flat for two consecutive frames. The final, desired solution to (10) is

$$D = D_A(N, M) \tag{14a}$$

$$\bar{D} = D_A(N, M)/N. \tag{14b}$$

Thus the DTW algorithm requires on the order of NM distance calculations, and NM sets of combinatorics [(12) and (13)] to obtain the best path and the total distance for each reference pattern. We now consider alternative path finding techniques which seek to reduce the number of local distance calculations.

III. An Ordered Graph Search Approach to Finding the Best Path Through a Grid

We have shown that the conventional DTW algorithm solves the problem of optimally time aligning a test and a reference pattern, at the same time providing a measure of the similarity (distance) between test and reference along the alignment path. By restructuring the entire time alignment problem as a problem in finding the best path through a finite grid of points, we can take advantage of a large class of ordered tree and graph searching algorithms, as described by Nilsson [8], [9], to find the best path with substantially reduced computation of local distances. These searching algorithms have been commonly used in the artificial intelligence (AI) area for machine simulation of cognitive processes, e.g., problem solving or game playing.

The way in which we apply graph searching to dynamic time warping is illustrated in Fig. 2. We represent the grid of points in which the alignment path can lie as a directed graph in which the nodes represent local distances (between test and reference frames), and allowable node transitions are represented by branches of the directed graph (digraph). An

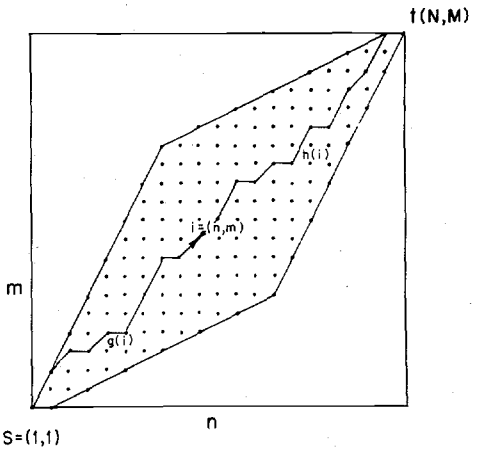


Fig. 2. Illustration of the nodal structure and a typical path for ordered graph searching.

ordered graph searching (OGS) algorithm is then applied to find the best time alignment path.

Before describing the graph searching algorithm, it is important to understand why such a procedure can lead to significant reduction in computation over standard DTW algorithms. This gain in efficiency for the digraph search is achieved by omitting the local distance calculations associated with nodes which are not searched. Dynamic programming, on the other hand, requires that *all* local distances within the global constraints be calculated. By restricting the digraph search to investigate only the most likely warping paths, the number of nodes that are expanded (i.e., used in subsequent calculations) can be kept to between $\frac{1}{3}$ and $\frac{1}{2}$ that used for the DP search; at the same time, under certain readily attainable conditions, the resulting warping path can be shown to be optimal, i.e., identical to the one found by the DP algorithm.

A. The Directed Search Algorithm

Consider the graph structure of Fig. 2. Each point in the grid is a node, and we designate the ith node by its coordinates (n, m), i.e.,

$$i = (n, m). \tag{15}$$

We denote the starting node as $s = (1, 1)$, and the ending node as $t = (N, M)$.

For any path through the grid passing through node i, we denote the path cost (the accumulated distance along the path) as

$$f(i) = g(i) + h(i) \tag{16}$$

where $g(i)$ is the minimal cost of the path from node s to node i, and $h(i)$ is the minimal cost of the path from node i to node t.

For a directed search through the grid (i.e., proceeding from left to right), the cost $g(i)$ along the path to node i is known exactly; however, the cost $h(i)$ from node i to node t is not known, and therefore must be estimated. Thus, an estimate of a minimal cost path passing through node i is

$$\hat{f}(i) = g(i) + \hat{h}(i) \tag{17}$$

where $\hat{h}(i)$ is the estimate of $h(i)$. Thus, in trying to find the minimal cost path through the grid, we build up a series of nodes for which we know the exact cost from the start node, and for which we estimate a cost to the terminal node. We expand the node which currently provides the smallest cost estimate (simultaneously keeping track of all previously encountered nodes, for backtracking) until we reach the terminal node t.

It should be clear that any path from start node s to intermediate node i is completely characterized by the nodal state, which includes:

1) the node coordinates (n, m);

2) the estimated cost $\hat{h}(i)$ from the node i to the end;

3) a pointer to the previous node on the path (the ancestor or parent node);

4) the true cost $g(i)$ from the start node to node i—the cost $g(i)$ is saved to facilitate computation of $\hat{f}(i')$ where i' is a successor node to i.

In solving for the minimal cost path through the grid, generally nodes from several paths are encountered. By using the nodal state information above, an "open" list of potential paths is maintained where each list entry is the nodal state of the last node on the path. The open list is sorted so that the last node of the path having the lowest estimated total cost $\hat{f}(i)$ is at the top of the list. The algorithm tries to find the minimal cost path through the grid by removing the top node from the open list (saving the nodal state on a "closed" list), and "expanding" it to generate all legal (within the local path constraints) successor nodes. New path cost estimates are computed for each of these successor nodes and they are sorted into the open list. The process starts with only the start node s on the open list and continues until a path to the terminal node t is found. Generally, a terminal node t is not identified as terminal until an expansion of t is attempted and no successors are generated. (However, in this implementation the terminal node is identified before it is added to the open list.) The warping path may then be found by tracing backward from node t to node s by using the parent node pointer information saved during the searching process. (For most isolated word recognition applications the warping path is not required and this trackback procedure can be omitted.)

The path that first terminates on node t will be minimal cost (optimal) if the following conditions are met:

1) the expansion operation is consistent for all nodes;

2) $g(i) > 0 \; \forall \, i \neq s$ and $g(i)$ is monotonic;

3) $\hat{f}(i) \leqslant f(i) \; \forall \, i$;

4) $\hat{h}(i)$ is monotonic for all potential paths $\hat{h}(i) > 0, \forall \, i \neq t$.

To show that these conditions are sufficient to find the optimal path we have to show that the search is guaranteed to reach the terminal state t, and having reached this state, the resulting path is minimal cost. As proved by Nilsson [9, pp. 74-79] for finite graphs the search must always terminate since these are only a finite number of nodes that can be expanded. To show that the search finds the optimal path to t, we assume that a path to node t gets a nonminimal cost $f(t) = g(t) > f^*(t)$ where $f^*(t)$ is the minimal cost to node t. By condition 3) (i.e., the estimator of cost is less than or equal to the true cost) there existed, before expansion of node t (termination) a node i' such that $\hat{f}(i') \leqslant f^*(t) < f(t)$. This implies that this node would have been expanded before t, contradicting the assumption that the search terminated. Thus, conditons 1)-4) are adequate to guarantee that a minimal cost path from s to t exists and can be found.

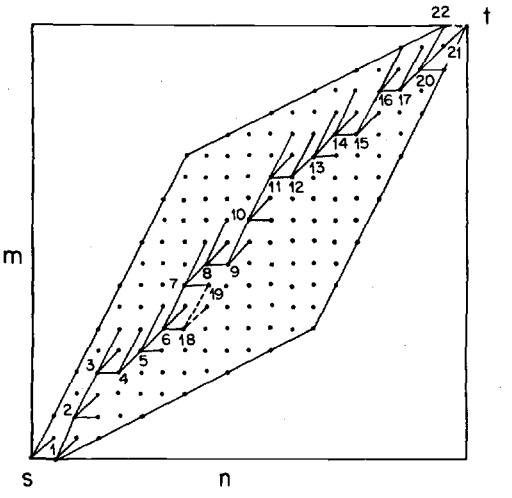


Fig. 3. Illustration of the computation to determine a path from node s to node t, using the ordered graph searching concept.

A similar argument can be made for each node in the graph. Upon first expanding node i, an optimal path has already been found from s to i. A useful consequence of this is that, upon expanding a node, if any successor node generated appears on the closed list, it need not be added to the open list again since an optimal path to this node was already found. Furthermore, if the successor node already appears on the open list it need not be added again since the path by which this node was previously generated is optimal to the expanded node and must be of lower cost than the path currently under consideration, or else it would not have been expanded previously.

Fig. 3 illustrates how the digraph search would work on a constrained grid of points representative of the one conventionally used in DTW applications. Initially the search is at node s which is expanded to three possible successor nodes. Node 1, having the smallest distance $\hat{f}$, is next expanded to its two possible successor nodes (recall that the path is constrained to rise by 1 every other frame); the unexpanded nodes being retained on the open list. Node 2, again with the smallest $\hat{f}$ is next expanded. This process continues until node 17, which, upon expansion, shows node 18 (an earlier successor) to have smallest $\hat{f}$ of all open nodes. Upon expanding node 18, successor node 19 is generated but not added to the open list since it was encountered when node 7 was expanded previously. Node 18 and its successors are expanded until node 20 comes to the top of the open list and this local search is terminated by the optimality property just described. Node 20 is next expanded to node 21, followed by node 22 and finally state t is reached with a minimal cost path.

The computational savings, for this simple example, over a convention DP approach are approximately the ratio of grid points to the number of nodes for which distances are calculated, or about 3:1. That such savings are achievable in real examples will be shown later in this paper.

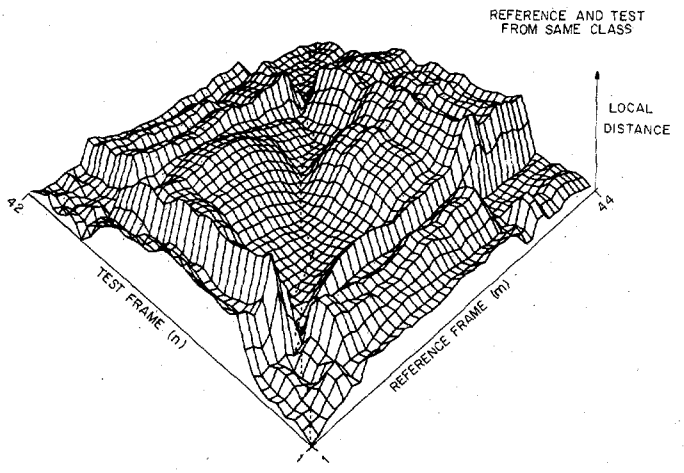


Fig. 4. Plots of local distance in the (n, m) plane for reference and test patterns from the same word class.

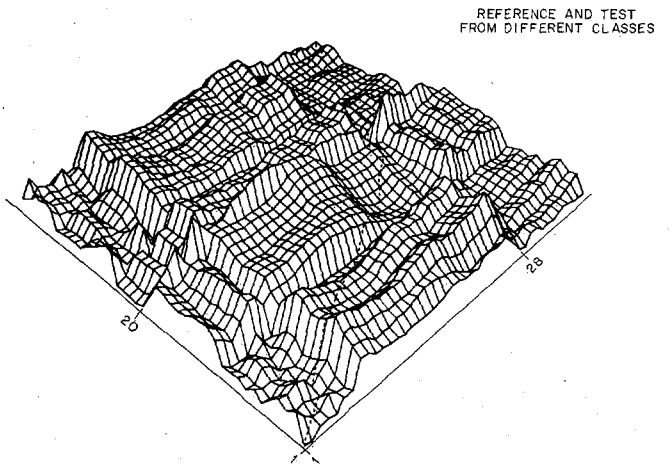


Fig. 5. Plots of local distance in the (n, m) plane for reference and test patterns from different word classes.

B. Ordered Graph Searching Applied to DTW

As applied to DTW algorithms for isolated word recognition, all the optimality conditions of Section III-A are satisfied, *except* condition 1). This condition is violated because the local constraints are data dependent, i.e., the optimal path cannot stay flat for two consecutive frames, and thus the expansion of any node depends on the path to that node. This means that for ordered graph searching (as for conventional DTW algorithms [5]) optimality of the path is not guaranteed. However, as shown by Myers *et al.* [5], the nature of the problem essentially makes the path finding a robust procedure which is basically insensitive to the data dependent path constraints, especially when the test and reference patterns are from the same word class. This point is illustrated in Figs. 4 and 5 which show plots of the local distance $\tilde{d}(T(n), R(m))$ over the entire (n, m) plane for reference and test from the same class (Fig. 4 for the digit 5), and for reference and test from different classes (Fig. 5 for the digits 5 and 6). Also shown in these figures is the optimum warping path (indicated by the dashed line). As shown in Fig. 4 the warping path lies in a valley in the local distance function. This valley is reasonably broad along most of the path; hence slight deviations from the time optimal path do not generally result in substantial increases in path cost (distance).

When the reference and test are from different classes (Fig. 5), the warping path generally deviates substantially from the diagonal path, indicating highly nonlinear compressions and expansions of the time scales to achieve best matches. The general shape of the distance function is a series of sharp peaks and valleys which fluctuate rapidly in the (n, m) plane. Thus, small deviations in the warping path, due to the local constraints, can lead to significant cost increases over the optimal path.

If the previous description were entirely correct, we would be able to take advantage of it to increase the separation between the distance (cost) distributions of "same" and "different" words, since the calculated distances for same words are essentially minimal costs, whereas for different words they are above minimal cost. Although in practice this situation does occur, the magnitude of the effect is small. The key point, however, is that paths and path distances obtained by the OGS approach are comparable to those obtained from DP algorithms.

1) Flowchart of OGS Procedure: A flow diagram of the basic ordered graph searching algorithm is given in Fig. 6. The start node $s = (1, 1)$ is the only node on the open list and the closed list is empty at the beginning of the search. The open list is always maintained as a sorted list of nodes such that the node having least estimated total warp path cost heads the list. This head node is removed from the open list for expansion into successor nodes. Expansion proceeds by generating one node at a time until all possibilities are exhausted. For efficiency reasons, an array of flags is maintained which indicates which nodes have been generated. In this way, the step in the flow labeled "check OPEN & CLOSED" can be performed by simply checking one element of the array rather than searching two lists. The generated node must lie within the global constraints. If it is not the terminal node, then $\hat{f}(i) = g(i) + \hat{h}(i)$ is computed. Using this $\hat{f}(i)$ value the node is inserted in the open list at the appropriate location and the process continues.

When the terminal node is found, the $g(i)$ already calculated for this node is the path cost $g(t) = f(t)$. The warping path may be recovered if necessary by following the parent node pointers backward from the terminal node to the start node and the solution is complete.

2) The Estimator Function $\hat{h}(i)$: The only unspecified quantity for the OGS algorithm is the estimator function $\hat{h}(i)$ used to provide the estimated path cost $\hat{f}(i) = g(i) + \hat{h}(i)$. The quantity $\hat{f}(i)$ must be calculated at each node i visited during the search process, and since $g(i)$ is known exactly, only $\hat{h}(i)$ must be specified to give $\hat{f}(i)$.

There are several ways that $\hat{h}(i)$ could be calculated. Since $\hat{h}(i)$ is the distance (cost) along the path from node i to the terminal node t, and since $\hat{h}(i)$ must underestimate the true path cost $h(i)$ (to satisfy the path optimality constraints of Section III-A), then for $i = (n, m)$ we have

$$\hat{h}(i) \leqslant h(i) = \sum_{k=n+1}^{N} \tilde{d}(T(k), R(w(k))). \tag{17}$$

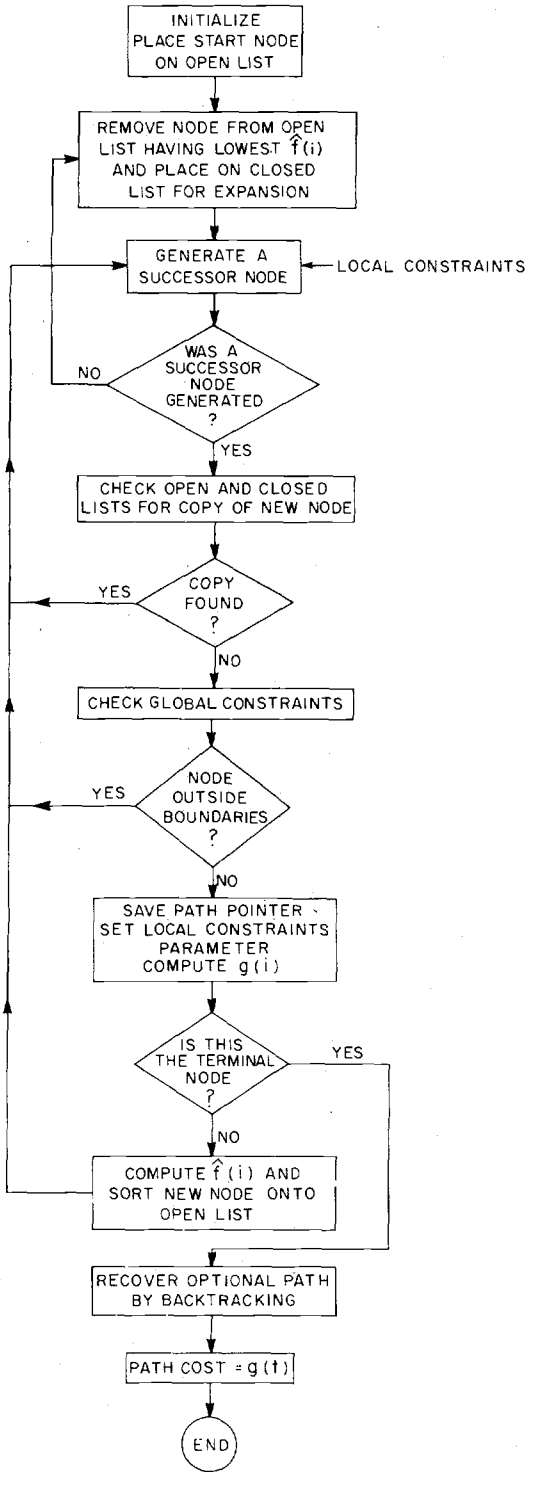


Fig. 6. Flowchart of the OGS method.

Equation (17) says that the true path cost from node i to node t is the sum of the local distances along all the nodes in the path, and for the asymmetric path constraints used in the DTW implementation, this distance is the sum of the distances along the $(N - n)$ grid points of the path. When T and R are from the same word class, then, along the optimal alignment path, we have the theoretical result that

$$P_{\tilde{d}(T(k),R(w(k)))}(\beta) = F(\beta), \tag{18}$$

i.e., the probability density function of $\tilde{d}$ is independent of T, R, and k [10]. For example, for the LPC parameter set used here we have

$$P_{\tilde{d}}(\beta) = \chi^2(\beta). \tag{19}$$

Based on the above discussion we can use, as a bound on $\hat{h}(i)$, the quantity

$$\hat{h}(i) = (N - n) \cdot \bar{d} \tag{20}$$

where $\bar{d}$ is sufficiently small so that we can guarantee that the probability that $\hat{h}(i) > h(i)$ is kept to any desired value.

The problem with the estimator of (20) is that for test and reference words of different word classes, the estimator is a gross underestimator, causing a needlessly large number of nodes to be searched. For such comparisons we would prefer a gross overestimator; whereas for cases when reference and test are from the same word class we need the underestimator of (10). To combat these difficulties we have developed an adaptive estimation procedure, which we now describe.

Consider first a fixed estimator of the form

$$\hat{h}(i) = (N - n)\,\alpha \tag{21}$$

where α is a parameter of the estimator. Fig. 7 shows some representative plots of three measures of computation and accuracy, namely

1) accumulated path cost $f(t)$,
2) number of distance calculations N_D, and
3) number of nodes expanded N_E,

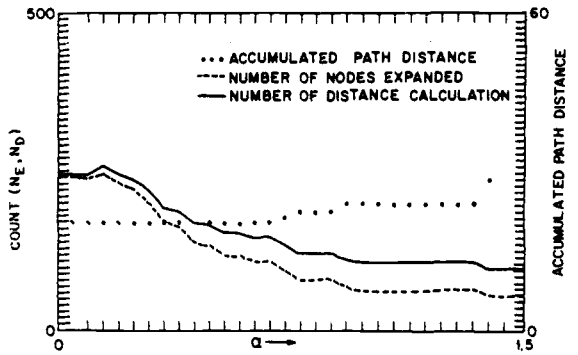
as a function of α. Fig. 7(a) is for a typical case when reference and test words are from the same class, and Fig. 7(b) is for a typical case when reference and test words are from different classes. It can be seen from Fig. 7(a) that for values of $\alpha \leqslant 0.885$, the accumulated path cost remains constant, whereas N_D and N_E fall dramatically as α increases above 0.1. For this example the average cost per frame was 0.45, showing that α can increase above the value by almost a factor of 1.7 before optimality of the path is sacrificed. Other examples have shown similar behavior as a function of α.

For the data of Fig. 7(b), when reference and test words were different, decreases of N_E and N_D became significant only for very large values of α, e.g., $\alpha > 1.2$, whereas path cost is seen to be almost independent of α.

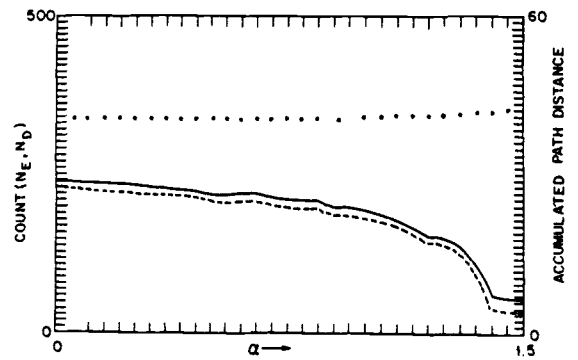
The above results suggest a data adaptive estimator of the form

$$\hat{h}(i) = (N - n)\,\alpha \frac{g(i)}{n}. \tag{22}$$

The estimator of (22) has the advantage that for words of the same class, $g(i)/n$ eventually becomes small, giving an effective α multiplier in the correct range, whereas for words in different classes, $g(i)/n$ eventually becomes large, giving the best results here. Experimentation with the estimator of (22) showed that the resulting path costs were basically insensitive to α over a wide range of α. Based on this experimentation, a value of $\alpha = 0.7$ was chosen. Although there is no theoretical guarantee that $\hat{h}(i)$ of (22) is an underestimate of $h(i)$ in all



(a)



(b)

Fig. 7. Plots of N_D, N_E, and $\overline{D}$ versus α for reference and test patterns in (a) the same class and (b) in different classes.

cases, practical experience indicates this is essentially always the case.

Efficiency can be improved further (without significant loss in accuracy) by forcing α to become much larger than 0.7 when preliminary results indicate that the test and reference words come from different classes. This makes $\hat{h}(i)$ predominant in calculating $\hat{f}(i)$ and tends to minimize backing up during the search. In this case the actual warp path cost is not important as long as it is large. The algorithm can predict early in the search if the test and reference do not belong to the same class by observing the average per test frame value of $g(i)$. A large value for $g(i)$ indicates a potential mismatch. There is always the possibility that for the first few test frames, $g(i)$ will be larger than the global average. This could cause the algorithm to predict a class mismatch during the first few frames of the search. Since the sensitivity of path cost to α is low, however, this will not generally cause any substantial difference in total path cost as long as the class mismatch prediction is corrected at each frame.

Based on the above discussion, the final form of the estimator function is

$$\hat{h}(i) = \begin{cases} 0.7\dfrac{g(i)}{n}(N-n) & \text{if } \dfrac{g(i)}{n} < \beta \qquad (23a) \\ 2.0\dfrac{g(i)}{n}(N-n) & \text{if } \dfrac{g(i)}{n} > \beta. \qquad (23b) \end{cases}$$

Values of β between 0.6 and 0.7 were used in two evaluation tests to be described in Section IV. The chosen value of β is essentially the largest reasonable average distance one would expect to encounter when comparing test and reference words of the same class.

IV. Experimental Comparison of DP and OGS

To compare the performance of the OGS algorithm of Section III with the standard DTW algorithm, two recognition experiments were performed. For the first experiment, each of 6 talkers (3 male, 3 female) trained the recognizer on isolated digits (using a robust training method [11]), thereby obtaining 6 sets of speaker trained templates. Then each talker spoke the 10 digits 5 times to form a test set of 50 utterances.

For the second experiment a speaker independent set of templates was used in which each word of the vocabulary was represented by 6 templates. The vocabulary for this experiment was 129 word airlines vocabulary [12], and the templates were obtained from a clustering analysis of the speech of 100 talkers (50 male, 50 female) [13]. A set of 4 talkers (2 male, 2 female—all not part of the training set) was used for this experiment, with each talker saying the vocabulary one time.

Both recognition experiments used speech recorded off a dialed-up telephone line. The "standard" LPC based recognizer of Rabiner *et al.* was used to derive the features of the speech, and to provide a set of Q best recognition candidates [14]. The normalize-and-warp procedure [5] was also used to provide fixed length test and reference patterns. For each experiment both the DP and the OGS algorithms were used, and the results are compared in terms of recognition accuracy and computation.

A. Results of Experiment 1—Speaker Trained Recognition Using Isolated Digits

The results of the first experiment on the digit vocabulary are presented in Table I and in Fig. 8. Table I gives statistics, for each talker, for both the OGS and DP algorithms, for the following:

1) number of local distance calculations N_D;
2) average distance to the correct reference $\overline{D}$;
3) average distance to the second candidate $\overline{D}_2$;
4) recognition error rate for the top candidate;
5) recognition error rate for the top two candidates.

Fig. 8 shows histograms (obtained from one of the 6 talkers), as a function of comparisons with the correct reference (C), and with all incorrect references (I), of

1) the number of nodes expanded N_E;
2) the number of local distance calculations N_D;
3) the average per frame distance from the optimal path $\overline{D}$.

(Histograms from each of the 6 talkers showed similar behavior).

Examination of Table I shows the following.

1) For correct references an average reduction in the number of local distances by a factor of 3.2 is obtained.

2) For incorrect references, an average reduction in the number of local distances by a factor of 2.4 is obtained.

TABLE I
RECOGNITION STATISTICS FOR THE 10 WORD DIGIT VOCABULARY (SPEAKER TRAINED)

		Talker Number/Sex						
		1(M)	2(F)	3(F)	4(M)	5(M)	6(F)	Average
Number DP Distances		620	620	620	620	620	620	620
Number OGS Distances/C		172	226	162	230	212	167	195
Number OGS Distances/I		257	268	248	256	267	240	256
Average Distance To Correct Reference	DP	.41	.40	.36	.34	.36	.17	.34
	OGS	.43	.42	.37	.35	.38	.17	.35
Average Distance To Second Candidate	DP	.66	.66	.60	.50	.56	.50	.58
	OGS	.68	.70	.62	.52	.59	.52	.61
Error Rate (%) Top Candidate	DP	0	4	0	6	10	0	3.33
	OGS	2	4	0	8	12	0	4.33
Error Rate (%) Top 2 Candidates	DP	0	0	0	0	0	0	0
	OGS	0	0	0	2	2	0	0.67

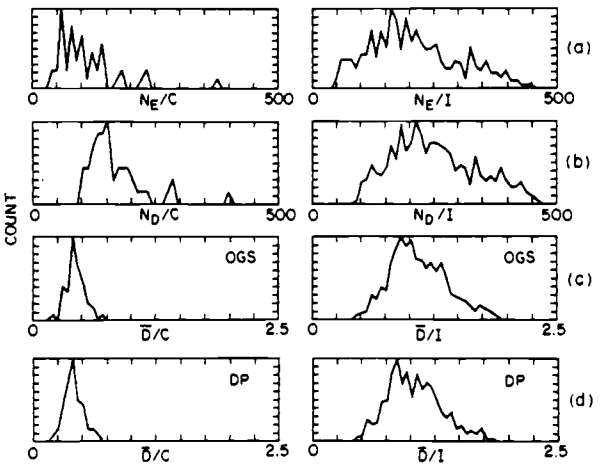


Fig. 8. Histograms of N_E, N_D, and $\bar{D}$ conditioned on correct and incorrect references for the digit recognition experiment.

3) The average distance of the OGS method to correct references (0.35) is slightly larger than the average distance of the DP method to correct references (0.34); similarly the average distance of OGS to the second candidate (0.6) is slightly larger than the DP distance (0.58).

4) The average error rate for the OGS system is slightly larger (for both top 1 and top 2 candidates) than for the DP system.

The results above indicate that the OGS method yields only slightly worse accuracy results than the DP method, at the same time achieving about a 2.5 : 1 overall reduction in distance computation.

Examination of the histograms of Fig. 8 shows a fairly wide spread in the N_E and N_D statistics for incorrect references, indicating that for some words a large percentage of the available nodes were used. For correct references a lower, tighter distribution is found for both N_E and N_D. Finally, it is seen that the overall distance distributions for DP and OGS [Figs. 8(c) and (d)] are similar, and the small differences reflect the loss in accuracy of not always finding the optimal warping path.

TABLE II
RECOGNITION STATISTICS FOR THE 129 WORD AIRLINE VOCABULARY (SPEAKER INDEPENDENT)

		Talker Number/Sex				
		1(M)	2(M)	3(F)	4(F)	Average
Number DP Distances		682	682	682	682	682
Number OGS Distances/C		242	243	240	249	244
Number OGS Distances/I		268	260	266	260	264
Average Distance To First Candidate	DP	.257	.259	.286	.301	.276
	OGS	.262	.264	.292	.306	.281
Average Distance To Second Candidate	DP	.319	.309	.329	.347	.326
	OGS	.327	.318	.337	.357	.335
Error Rate (%) Top Candidate	DP	10.8	7.8	15.5	19.4	13.4
	OGS	10.8	10.1	14.7	20.2	14.0
Error Rate (%) Top 2 Candidates	DP	2.3	5.4	6.2	9.3	5.8
	OGS	2.3	5.4	7.8	13.2	7.2

B. Results of Experiment 2–Speaker Independent Recognition of Airline Terms

The results of the speaker independent test, using 6 templates per word and a 129 word airlines vocabulary are given in Table II and Fig. 9. The type of statistics given in the table and Fig. 9 are similar to those of Table I and Fig. 8, with one exception. Since there were 6 "correct" references for each spoken word, we present statistics on the first candidate (in Table II) and on both the correct references and the first candidate in Fig. 9.

The results on reduction in distance computation are similar to those of Experiment 1. However here the reduction for correct references is about 2.8 : 1, and for incorrect references it is 2.6 : 1, thereby indicating a somewhat larger overall reduction than for the digit vocabulary.

Another difference from Experiment 1 is that the average distance of the second candidate is close to the average distance of the first candidate, indicating that, in general, the first two candidates were reference patterns from the correct word class.

As in the digit experiment, it was found that the average separation of the distributions of distances for correct and incorrect references for the OGS method, was slightly larger than for the DP method. The differences, however, occur primarily on the upper side of the distributions for the incorrect references, thereby having little effect on recognition accuracy.

Close scrutiny of the OGS distribution reveals slight but consistent irregularity in the distance distributions for incorrect references. This irregularity can be attributed to the underestimator to overestimator function switching, which occurs when predicted distance per test frame exceeds the chosen value of $\beta = (0.7)$. When the estimator function $\hat{h}(i)$ becomes a cost overestimator it generally forces the actual

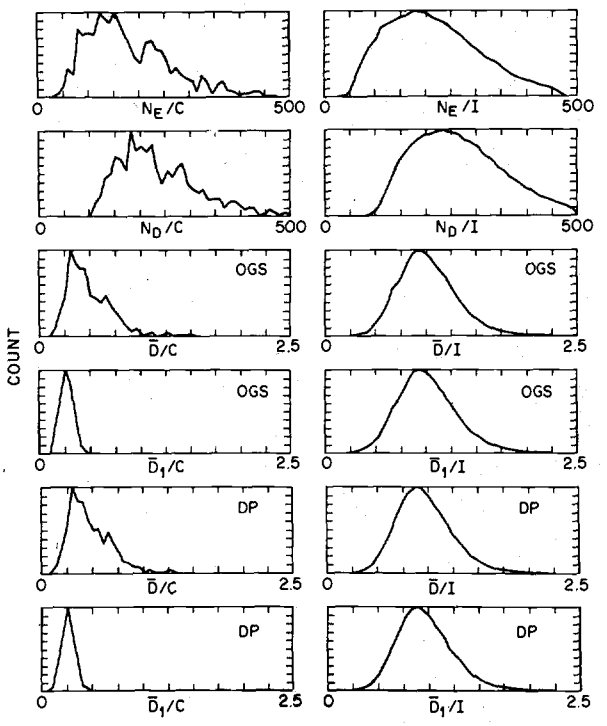


Fig. 9. Histograms of N_E, N_D, $\bar{D}$, and $\bar{D}_2$ conditioned on correct and incorrect references for the airlines vocabulary recognition experiment.

warp path cost to increase slightly thus forming a slight depression in the distribution of distances just above a distance value of 0.7. This irregularity occurs at sufficiently high distance values as to have no effect on recognition.

Finally, we see that the average recognition error rates on the top candidate are comparable with the DP method yielding a 0.6 percent smaller error rate than the OGS method. Thus, as in the digit experiment, we see that a 2.6:1 reduction in local distance computations can be achieved at the expense of about 0.6 percent in recognition accuracy.

V. Discussion

The purpose of this paper was to describe an alternative approach to dynamic programming to solve the time alignment problem for isolated word recognition. It was shown that the class of ordered graph searching methods, widely used in the area of artificial intelligence, could readily be applied to the time alignment problem. One such algorithm was developed and discussed in detail.

It was shown that the OGS method could solve the time alignment problem with essentially the same accuracy as DP methods; however the required computation for local distances was reduced by a factor of about 2.5.

There are, however, at least two mitigating circumstances that affect the results presented above. First is that there are alternative ways of achieving computational reductions in standard DP approaches. For example Rabiner *et al.* [14] have proposed the use of a rejection threshold to monitor the DTW distance as it proceeds through the grid and to abort any reference pattern whose accumulated distances exceeds the rejection threshold. This rejection threshold can either be static or dynamic, depending on detailed knowledge of the expected accumulated distance histogram. Calculations by Rabiner *et al.* show that potential savings of from 50–75 percent of the computation can be achieved in this manner, with no loss in recognition accuracy.

It should be clear to the reader that the rejection threshold idea can equally well be applied to the OGS method as to the DP method; hence the data reduction of the OGS method is essentially in addition to that of the rejection threshold.

A second, and perhaps more substantial objection to the OGS method is that the reduction in distance computation is attained at the expense of a more complicated control structure. This combinatoric effort has been estimated to be about 200–250 percent of the combinatoric effort of the DP algorithm based on CPU time measurements. This increased overhead may have a substantial impact on the overall efficiency, especially if most of the numerical effort can be handled by high speed hardware.

If we assume a typical microprocessor application without special purpose hardware, nominal time to perform one DTW is about 5 s. Typically, combinatorics consumes only about 0.1 s of this time leaving 4.9 s for numerical computation using standard DP methods. The OGS algorithm is well suited to this type of application. Combinatoric time for OGS would be about 0.2–0.25 s while numerical computation time would drop to about 1.7 s. Thus the OGS method would be more than 60 percent faster than the DP method, performing a typical DTW in less than 2 s.

On the other hand, special purpose hardware now exists for computing local distances in a parallel manner. Under there circumstances the ratio of numeric to combinatoric time may be considerably less than one. The OGS method would then be considerably slower than the standard DP method.

VI. Summary

In summary, the OGS method has been shown to be a practical alternative to standard DP methods for solving the DTW problem. The OGS method substantially reduces numerical computation at the expense of increased combinatoric effort without significantly altering the DTW path cost. OGS is a particularly viable technique for microprocessor implementations of speech recognition systems.

References

[1] G. M. White and R. B. Neely, "Speech recognition experiments with linear prediction, bandpass filtering, and dynamic programming," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 183–188, Apr. 1976.

[2] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 67–72, Feb. 1975.

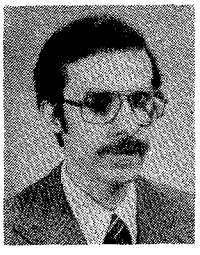
[3] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, pp. 43–49, Feb. 1978.

[4] L. R. Rabiner, A. E. Rosenberg, and S. E. Levinson, "Considerations in dynamic time warping for discrete word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, pp. 575–582, Dec. 1978.

[5] C. S. Myers, L. R. Rabiner, and A. E. Rosenberg, "Performance tradeoffs in dynamic time warping algorithms for isolated word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 622–633, Dec. 1980.

[6] H. F. Silverman and N. R. Dixon, "State constrained dynamic

programming (SCDP) for discrete utterance recognition," in *Proc. Int. Conf. ASSP*, Apr. 1980, pp. 169–172.
[7] M. H. Kuhn, H. Tomaschewski, and H. Ney, "Fast nonlinear time alignment for isolated word recognition," in *Proc. Int. Conf. ASSP*, Apr. 1981, pp. 736–740.
[8] N. J. Nilsson, *Problem-Solving Methods in Artificial Intelligence.* New York: McGraw-Hill, 1971.
[9] —, *Principles of Artificial Intelligence.* Tioga, 1980.
[10] L. R. Rabiner and J. G. Wilpon, "A two-pass system for isolated word recognition," *Bell Syst. Tech. J*, vol. 60, pp. 739–766, May–June 1981.
[11] —, "A simplified, robust training procedure for speaker trained, isolated word recognition systems," *J. Acoust. Soc. Amer.*, vol. 68, pp. 1271–1276, Nov. 1980.
[12] S. E. Levinson, A. E. Rosenberg, and J. L. Flanagan, "Evaluation of a word recognition system using syntax analysis," *Bell Syst. Tech. J*, vol. 57, pp. 1619–1626, May–June 1978.
[13] J. G. Wilpon, L. R. Rabiner, and A. Bergh, "Speaker independent isolated word recognition using a 129 word airline vocabulary," to be published.
[14] L. R. Rabiner, S. E. Levinson, A. E. Rosenberg, and J. G. Wilpon, "Speaker independent recognition of isolated words using clustering techniques," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 336–349, Aug. 1979.



Michael K. Brown was born in Highland Park, MI, on January 4, 1951. He received the B.S.E.E. degree in 1973 in electrical engineering and the M.S. and Ph.D. degrees in 1977 and 1981, respectively, all from the University of Michigan, Ann Arbor.

From 1973 to 1976 he was employed with the Burrough's Corporation and was involved in the development of ink jet printer systems. From 1976 to 1980 he continued his work with Burrough's as a consultant while pursuing the Ph.D. degree at the University of Michigan. His dissertation was in the area of image processing and pattern recognition. Since 1980, he has been with the Speech Processing Group at Bell Laboratories, Murray Hill, NJ, where he has been involved in research in speech recognition and synthesis techniques.

Lawrence R. Rabiner (S'62–M'67–SM'75–F'75) was born in Brooklyn, NY, on September 28, 1943. He received the S.B. and S.M. degrees simultaneously in June 1964, and the Ph.D. degree in electrical engineering in June 1967, all from the Massachusetts Institute of Technology, Cambridge, MA.

From 1962 through 1964, he participated in the cooperative plan in electrical engineering at Bell Laboratories, Whippany, NJ, and Murray Hill, NJ. He worked on digital circuitry, military communications problems, and problems in binaural hearing. Presently, he is engaged in research on speech communications and digital signal-processing techniques at Bell Laboratories, Murray Hill. He is coauthor of the books *Theory and Application of Digital Signal Processing* (Prentice-Hall, 1975) and *Digital Processing of Speech Signals* (Prentice-Hall, 1978).

Dr. Rabiner is a member of Eta Kappa Nu, Sigma Xi, Tau Beta Pi, and a Fellow of the Acoustical Society of America. He is a former President of the IEEE S-ASSP Ad Com, and is currently a member of the S-ASSP Technical Committee on Digital Signal Processing, former member of the S-ASSP Technical Committee on Speech Communication, former Associate Editor of the S-ASSP TRANSACTIONS, member of the PROCEEDINGS OF THE IEEE Editorial Board, and a former member of the Technical Committee on Speech Communication of the Acoustical Society.