# Dynamic Time Warping for Isolated Word Recognition Based on Ordered Graph Searching Techniques

*M. K. Brown*

*L. R. Rabiner*

Bell Laboratories
Murray Hill, New Jersey 07974

*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 to 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 to 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 [2-4]. These procedures require calculation of the local distance between each possible reference and test frame (within a prescribed global range) in order to determine the optimal time alignment path relating reference and test frames.

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 modelling the DTW problem as one of finding a directed path through a constrained grid. By modelling the grid as a digraph (directed graph) 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%) can be achieved with only a small loss in accuracy.

## II. An Ordered Graph Search (OGS) Approach to Finding the Best Path through a Grid

A 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 [5], to find the best path with substantially reduced computation of local distances.

The way in which we apply graph searching to dynamic time warping is illustrated in Figure 1. 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 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 1/3 and 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.

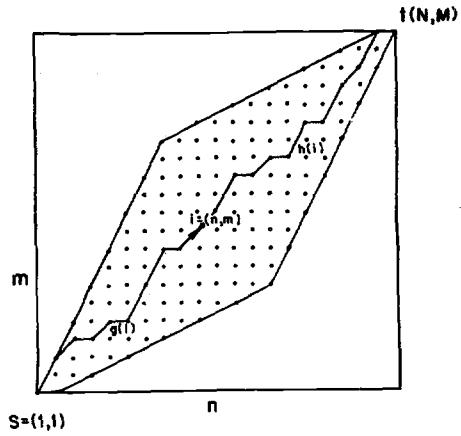


Fig. 1 Illustration of the nodal structure and a typical path for ordered graph searching.



CH 1746-7/82/0000 - 1255 $ 00.75 © 1982 IEEE

### 2.1 The Directed Search Algorithm

Consider the graph structure of Figure 1. Each point in the grid is a node, and we designate the $i^{th}$ node by its coordinates $(n,m)$, i.e.

$$i = (n,m) \tag{1}$$

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

$$f(i) = g(i) + h(i) \tag{2}$$

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

$$\hat{f}(i) = g(i) + \hat{h}(i) \tag{3}$$

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

It can be shown that the above conditions are sufficient to find the optimal path.

### 2.2 Ordered Graph Searching (OGS) Applied to DTW

As applied to DTW algorithms for isolated word recognition, all the optimality conditions of Section 2.1 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 [4]) optimality of the path is not guaranteed. However, as shown by Myers et al. [4], 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 Figures 2 and 3 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. 2 for the digit 5), and for reference and test from different classes (Fig. 3 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. 2 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).

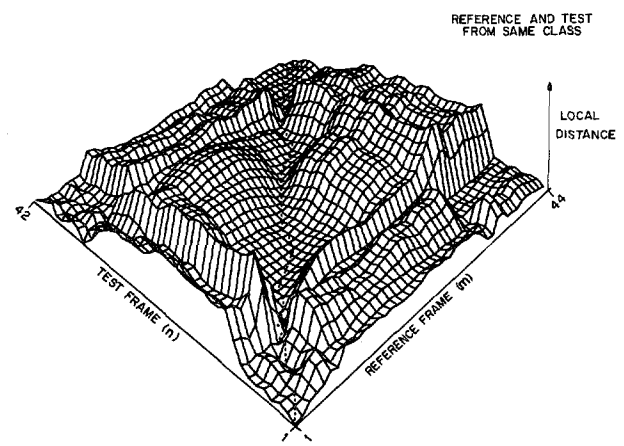


Fig. 2 Plots of local distance in the $(n,m)$ plane for reference and test patterns from the same word class.

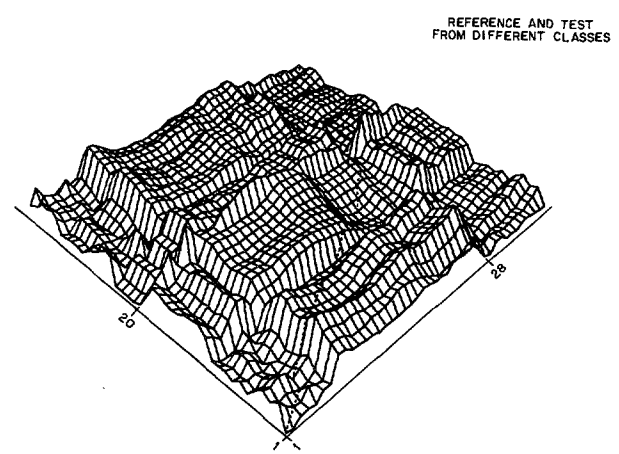


Fig. 3 Plots of local distance in the $(n,m)$ plane for reference and test patterns from different word classes.

When the reference and test are from different classes (Fig. 3), 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 above description were consistent, 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.

### 2.3 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 2.1), then for $i=(n,m)$ we have

$$\hat{h}(i) \leq h(i) = \sum_{k=n+1}^{N} \tilde{d}(T(k), R(w(k))) \tag{4}$$

Eq. (4) 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

$$P_{\tilde{d}(T(k),R(w(k)))}(\beta) = F(\beta) \tag{5}$$

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

$$P_{\tilde{d}}(\beta) = \chi^2(\beta) \tag{6}$$

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

$$\hat{h}(i) = (N-n) \cdot \bar{d} \tag{7}$$

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 Eq. (7) 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 an underestimator. To combat these difficulties we have developed an adaptive estimation procedure, which we now describe.

Consider first a fixed estimator of the form

$$\hat{h}(i) = (N-n)\alpha \tag{8}$$

where $\alpha$ is a parameter of the estimator. Figure 4 shows some representative plots of 3 measures of computation and accuracy, namely:

1. Accumulated path cost, $f(t)$

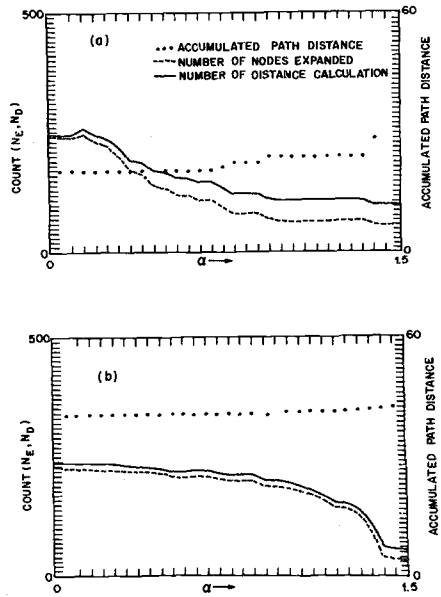


Fig. 4 Plots of $N_D$, $N_E$, and $\bar{D}$ versus $\alpha$ for reference and test patterns in the same class (4a) and in different classes (4b).

2. Number of distance calculations, $N_D$
3. Number of nodes expanded, $N_E$

as a function of $\alpha$. Figure 4a is for a typical case when reference and test words are from the same class, and Figure 4b is for a typical case when reference and test words are from different classes. It can be seen from Fig. 4a that for values of $\alpha \leq 0.885$, the accumulated path cost remains constant, whereas $N_D$ and $N_E$ fall dramatically as $\alpha$ increases above 0.1. For this example the average cost per frame was 0.45, showing that $\alpha$ can increase above this 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 $\alpha$.

For the data of Fig. 4b, when reference and test words were different, decreases of $N_E$ and $N_D$ became significant only for very large values of $\alpha$, e.g. $\alpha > 1.2$, whereas path cost is seen to be almost independent of $\alpha$.

The above results suggest a data adaptive estimator of the form

$$\hat{h}(i) = (N-n)\,\alpha\,\frac{g(i)}{n} \tag{9}$$

The estimator of Eq. (9) has the advantage that for words of the same class, $g(i)/n$ eventually becomes small, giving an effective $\alpha$ 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 Eq. (9) showed that the resulting path costs were basically insensitive to $\alpha$ over a wide range of $\alpha$. Based on this experimentation, a value of $\alpha = 0.7$ was chosen. Although there is no theoretical guarantee that $\hat{h}(i)$ of Eq. (9) is an underestimate of $h(i)$ in all cases, practical experience indicates this is essentially always the case.

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

$$\hat{h}(i) = \begin{cases} 0.7\,\dfrac{g(i)}{n}\,(N-n) \text{ if } \dfrac{g(i)}{n} < \beta & \text{(10a)} \\ 2.0\,\dfrac{g(i)}{n}\,(N-n) \text{ if } \dfrac{g(i)}{n} > \beta & \text{(10b)} \end{cases}$$

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

## III. 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), 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 a 129 word airlines vocabulary and the templates were obtained from a clustering analysis of the speech of 100 talkers (50 male, 50 female). 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 1 time.

Both recognition experiments used speech recorded off a dialed-up telephone line. The "standard" LPC based recognizer was used to derive the features of the speech, and to provide a set of $Q$ best recognition candidates. The normalize-and-warp procedure [4] 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.

### 3.1 Results of Experiments

The results of the first experiment on the digit vocabulary show 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.
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 to 1 overall reduction in distance computation.

The results of the speaker independent test, using 6 templates per word and a 129 word airlines vocabulary, are essentially the same as those on the digits vocabulary, namely a reduction in distance computation by about 2.6 to 1, with essentially no increase in error rate.

## IV. Discussion

It was shown in this paper 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 one can consider 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 show that potential savings of from 50 to 75% 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% 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 seconds. Typically, combinatorics consumes only about 0.1 seconds of this time leaving 4.9 sec 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 0.2-0.25s while numerical computation time would drop to about 1.7s. Thus the OGS method would be more than 60% faster than the DP method, performing a typical DTW in less than 2 seconds.

On the other hand, special purpose hardware now exists for computing local distances in a parallel manner. Under these 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.

### References


[1] G. M. White and R. B. Neely, "Speech Recognition Experiments with Linear Prediction, Bandpass Filtering, and Dynamic Programming", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-24, No. 2, pp. 183-188, April 1976.

[2] F. Itakura, "Minimum Prediction Residual Principle Applied to Speech Recognition", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-23, No. 1, pp. 67-72, Feb. 1975.

[3] H. Sakoe and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-26, No. 1, pp. 43-49, Feb. 1978.

[4] C. S. Myers, L. R. Rabiner, and A. E. Rosenberg, "Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-28, No. 6, pp. 622-633, Dec. 1980.

[5] N. J. Nilsson, *Problem-Solving Methods in Artificial Intelligence*, McGraw Hill, NY, 1971.