Performance Tradeoffs in Dynamic Time Warping Algorithms for Isolated Word Recognition

CORY MYERS, STUDENT MEMBER, IEEE, LAWRENCE R. RABINER, FELLOW, IEEE, AND AARON E. ROSENBERG, MEMBER, IEEE

Abstract—The technique of dynamic programming for the time registration of a reference and a test pattern has found widespread use in the area of isolated word recognition. Recently, a number of variations on the basic time warping algorithm have been proposed by Sakoe and Chiba, and Rabiner, Rosenberg, and Levinson. These algorithms all assume that the test input is the time pattern of a feature vector from an isolated word whose endpoints are known (at least approximately). The major differences in the methods are the global path constraints (i.e., the region of possible warping paths), the local continuity constraints on the path, and the distance weighting and normalization used to give the overall minimum distance. The purpose of this investigation is to study the effects of such variations on the performance of different dynamic time warping algorithms for a realistic speech database. The performance measures that were used include: speed of operation, memory requirements, and recognition accuracy. The results show that both axis orientation and relative length of the reference and the test patterns are important factors in recognition accuracy. Our results suggest a new approach to dynamic time warping for isolated words in which both the reference and test patterns are linearly warped to a fixed length, and then a simplified dynamic time warping algorithm is used to handle the nonlinear component of the time alignment. Results with this new algorithm show performance comparable to or better than that of all other dynamic time warping algorithms that were studied.

I. Introduction

TIME REGISTRATION of a test and a reference pattern is one of the fundamental problems in the area of automatic isolated word recognition. This problem is important because the time scales of a test and a reference pattern are generally not perfectly aligned. In some cases the time scales can be registered by a simple linear compression and expansion [1], [2]; however, in most cases, a nonlinear time warping is required to compensate for local compression or expansion of the time scale. For such cases, the class of algorithms known as dynamic time warping (DTW) methods have been developed. These algorithms have been shown to be applicable to the isolated word speech recognition problem and to greatly improve the accuracy of such systems [2]-[5].

Because of their importance to isolated word recognition systems, several investigations have been carried out to determine the "best" implementation of a dynamic time warping algorithm [6], [7]. Unfortunately, there has been no extensive evaluation in which all of the parameters of the DTW algorithms have been systematically varied to determine their effects on the performance of the recognizer. Thus, one of the purposes of this paper is to provide results on such a performance evaluation for a broad class of DTW algorithms. The second purpose of this paper is to unify the notation and mathematical framework for DTW algorithms for word recognition so that it is relatively easy to specify new variants of the algorithm, and to calculate their effects on basic parameters of the DTW method.

The organization of this paper is as follows. In Section II we give the mathematical framework for a general DTW algorithm and show how all previous formulations fit into this framework. In Section III we discuss the measures of performance used to compare the different DTW algorithms, namely, memory, speed of execution, and recognition accuracy. In Section IV we present results of a series of tests in which comparisons are made using each of the performance measures. Finally, in Section V, we discuss the implications of the results for isolated word recognition systems and present a modified algorithm for dynamic time warping which performs as well as or better than all the other algorithms tested.

II. Specification of the DTW Algorithm

We assume throughout this paper that, prior to the application of the DTW algorithm, the endpoints (beginning and ending frame) of the unknown isolated word (called the test) have been accurately located. We also assume that, via a conventional training process, the endpoints of each reference pattern (template) are accurately known. Hence, the problem of dynamic time warping can be formulated as a path finding problem over a finite grid as shown in Fig. 1. We denote the reference pattern as a sequence of frames, $R(n)$, $n = 1, 2, \cdots, N$, where $R(n)$ is, in general, a multidimensional feature vector that describes the characteristics of the nth frame of the spoken word. In Figs. 1 and 2, for the sake of illustration and clarity, we show $R(n)$ and $T(m)$ as one-dimensional functions of n and m. (Typically, a frame of data encompasses from 10-50 ms of data, and consecutive frames often overlap in time.) We denote the test pattern as a sequence of frames, $T(m)$, $m = 1, 2, \cdots, M$, where $T(m)$ is also a multidimensional feature vector.

Based on the model of Fig. 1, the DTW problem is to find an optimal path

$$m = w(n) \tag{1}$$

Manuscript received February 29, 1980; revised July 15, 1980. This work is based on the M.S. thesis of C. S. Myers, Massachusetts Institute of Technology, Cambridge, MA, February 1980.

C. Myers was with Bell Laboratories, Murray Hill, NJ 07974. He is now with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.

L. R. Rabiner and A. E. Rosenberg are with Bell Laboratories, Murray Hill, NJ 07974.

0096-3518/80/1200-0623$00.75 © 1980 IEEE

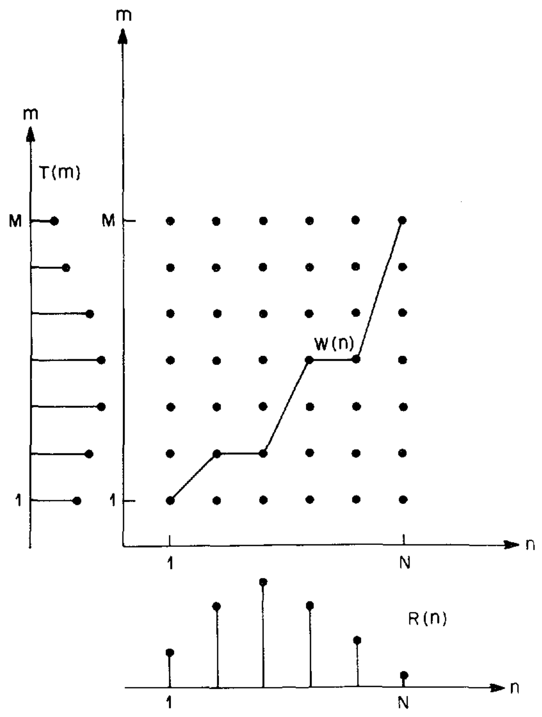


Fig. 1. An example illustrating the grid for warping $T(m)$ to $R(n)$ via the path $m = w(n)$.

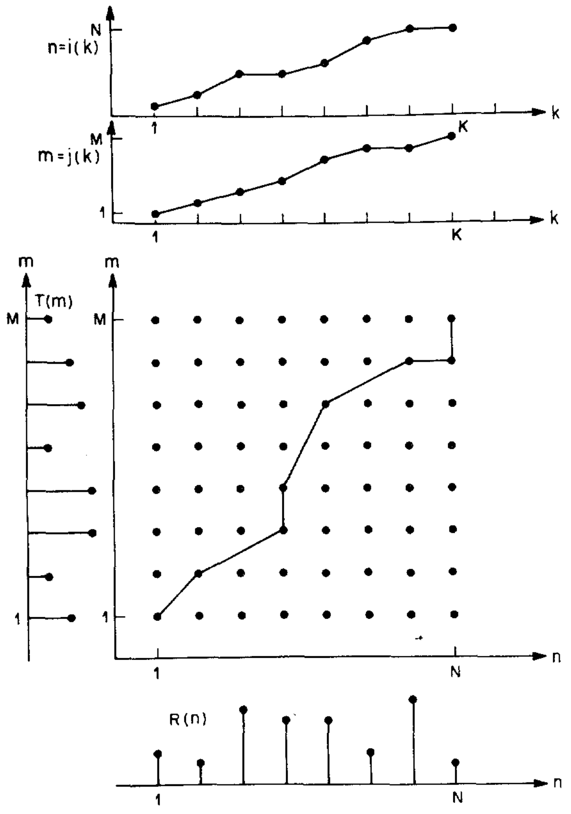


Fig. 2. An example of a parametric mapping of $T(m)$ to $R(n)$ via the intermediate time axis k.

in the (n, m) plane to minimize a total distance function D of the form

$$D = \sum_{n=1}^{N} \tilde{d}(R(n), T(w(n))) \tag{2}$$

where $\tilde{d}(R(n), T(w(n)))$ is the local distance between frame n of the reference pattern and frame $m = w(n)$ of the test pattern. A typical path $w(n)$ is shown in Fig. 1. It is important to note that $w(n)$ is restricted to begin at the point $n = 1, m = 1$, to pass through the grid of points (n, m) and to end at the point $n = N, m = M$.

In the formulation of (1) we have tacitly assumed that the path we are seeking can be expressed by a simple functional relation between m and n. It is not unreasonable, however, that the best warping path may not be functional [6]. In such cases, it is necessary to create a common time axis k, and to express both time axes (n and m) as functions of k, i.e.,

$$n = i(k), \quad k = 1, 2, \cdots, K \tag{3a}$$

$$m = j(k), \quad k = 1, 2, \cdots, K \tag{3b}$$

where K is the length of the common time axis. Fig. 2 shows a typical example in which $i(k)$ and $j(k)$ are shown as simple functions of k. The resulting curve in (n, m) space is shown in the middle of Fig. 2. It can be seen that the resulting curve is a monotonically increasing path from the point $(1, 1)$ to the point (N, M).

The formulation of (3) subsumes the formulation of (1) in that, if we choose the function $i(k)$ to satisfy the constraint

$$n = i(k) = k, \tag{4a}$$

then we get the result

$$m = j(k) = j(n) = w(n). \tag{4b}$$

In order to find the best path in the (n, m) plane, based on the parametric formulation of (3), several factors of the DTW algorithm must be specified, including:

1) endpoint constraints on the path,

2) local continuity constraints, i.e., the possible types of motion (e.g., directions, slopes) of the path,

3) global path constraints, i.e., the limitations on where the path can fall in the (n, m) plane,

4) axis orientation, i.e., the effects of interchanging the roles of the test and reference patterns,

5) distance measures, i.e., both the local distance measure and the overall distance metric used to determine the optimal path.

In the remainder of this section we discuss each of these factors and show how they affect the implementation (and performance) of the DTW algorithm.

A. Endpoint Constraints

For the case of isolated words with precisely determined endpoints for both the reference and test patterns (as assumed here), the parametric path endpoint constraints are of the form

$$i(1) = 1, \quad j(1) = 1, \qquad \text{beginning point} \tag{5a}$$

$$i(K) = N, \quad j(K) = M, \qquad \text{ending point.} \tag{5b}$$

B. Local Continuity Constraints

To further specify the optimal path, some local constraints must to be applied in order to guarantee that excessive compression or expansion of the time scales is avoided. A first constraint of this type is the monotonicity constraint, namely,

$$i(k+1) \geqslant i(k) \tag{6a}$$

$$j(k+1) \geqslant j(k). \tag{6b}$$

Next we would like to restrict the local range of the path in the vicinity of the point (n, m) as shown in Fig. 3(a). Here we assume that the only valid paths to the point (n, m) come from the points $(n-1, m-1)$, or $(n-1, m-2)$, or $(n-2, m-1)$. Hence, this set defines one possible set of local constraints. For this example, if we further assume that the path from $(n-2, m-1)$ must go through the intermediate point $(n-1, m)$, and the path from $(n-1, m-2)$ must go through the intermediate point $(n, m-1)$, we get the set of local constraints (which we call Type I constraints) shown in Fig. 3(b) [6].

Clearly, it is possible to specify a large number of sets of such local constraints for any particular problem. Thus, we need some formal method for specifying constraints of this type. One possible way would be to express the constraints as a set of productions in a regular grammar [8]. A production is a rule of the form

$$P_r \rightarrow (\alpha_1^{(r)}, \beta_1^{(r)})\,(\alpha_2^{(r)}, \beta_2^{(r)}) \cdots (\alpha_{L(r)}^{(r)}, \beta_{L(r)}^{(r)}) \tag{7}$$

where r signifies the rth production and $L(r)$ is the length of the rth production. The interpretation of the (α, β)'s as a production are that of the local changes used to reach a point in the path, i.e., to reach the point $i(k) = n$, $j(k) = m$, using a single production P_r, the path is traced backwards through $L(r)$ points as[1]

$$k\text{th point:}\quad i(k) = n, \qquad j(k) = m \tag{8a}$$

$$(k-s)\text{th point:}\quad i(k-s) = i(k) - \sum_{l=1}^{s} \alpha_l^{(r)} \tag{8b}$$

$$j(k-s) = j(k) - \sum_{l=1}^{s} \beta_l^{(r)} \tag{8c}$$

for $s = 1, 2, \cdots, L(r)$.

As an example, the set of productions for the Type I local constraints of Fig. 3 are

$$P_1^I \rightarrow (1,0)\,(1,1) \tag{9a}$$

$$P_2^I \rightarrow (1,1) \tag{9b}$$

$$P_3^I \rightarrow (0,1)\,(1,1). \tag{9c}$$

Using the productions of (9), an entire path from (N, M) to $(1, 1)$ can be expressed as a sequence of productions. By way of example, Fig. 4 shows a path specified by the productions (9)

$$P \rightarrow P_1^I P_1^I P_2^I P_1^I P_3^I P_2^I P_2^I.$$

The actual path may be obtained by substituting the definitions of the productions to yield the sequence

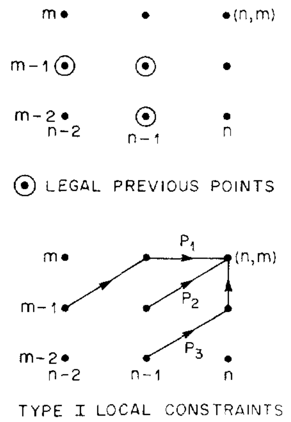


Fig. 3. An illustration of local path constraints to reach the point (n, m), and the resulting set of productions which describe the valid paths.

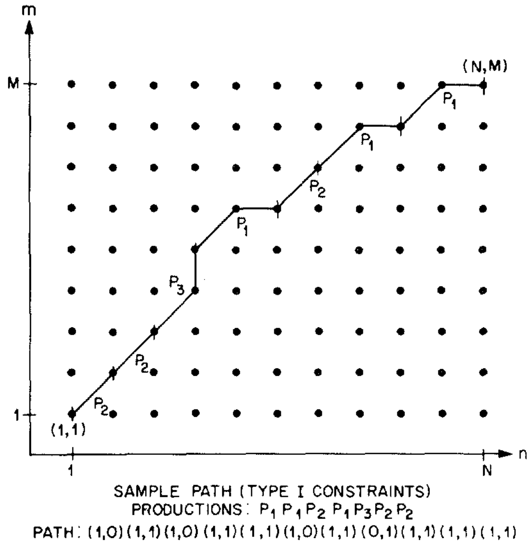

m
M
1
(N,M)
P1
P2
P3
(1,1)
1
N
n
SAMPLE PATH (TYPE I CONSTRAINTS)
PRODUCTIONS: P1 P1 P2 P1 P3 P2 P2
PATH: (1,0)(1,1)(1,0)(1,1)(1,1)(1,0)(1,1)(0,1)(1,1)(1,1)(1,1)

Fig. 4. A sample path and the set of productions for an illustrative time warping example.

$$P \rightarrow (1,0)\,(1,1)\,(1,0)\,(1,1)\,(1,1)\,(1,0)\,(1,1)\,(0,1)\,(1,1)\,(1,1)\,(1,1).$$

We see that this sequence specifies, tracing backwards, the path used to reach the point (N, M) from the point $(1, 1)$.

Using the concept of productions to specify the local path constraints, the monotonicity conditions (6) are readily given as

$$\alpha_l^{(r)}, \beta_l^{(r)} \geqslant 0 \qquad \text{for all } l. \tag{10}$$

Furthermore, simple expressions can be obtained for the maximum and minimum amount of expansion (or inversely compression) of the time scales directly from the productions. If we denote the maximum expansion as E_{MAX}, and the minimum expansion as E_{MIN}, then

$$E_{\text{MAX}} = \max_{(r)} \left[\sum_{l=1}^{L(r)} \beta_l^{(r)} \Big/ \sum_{l=1}^{L(r)} \alpha_l^{(r)} \right] \tag{11a}$$

$$E_{\text{MIN}} = \min_{(r)} \left[\sum_{l=1}^{L(r)} \beta_l^{(r)} \Big/ \sum_{l=1}^{L(r)} \alpha_l^{(r)} \right]. \tag{11b}$$

[1] The reader should recall that for DTW algorithms, the path is retrieved backwards from the end to the beginning since the optimal path is indeterminate until the end is reached.

LOCAL CONSTRAINTS

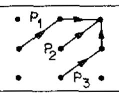
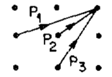
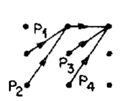
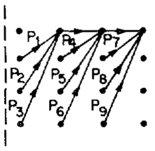
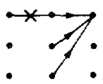
TYPE	PICTORIAL	PRODUCTIONS	E_{MAX}	E_{MIN}
I		$P_1 \to (1,0)(1,1)$ $P_2 \to (1,1)$ $P_3 \to (0,1)(1,1)$	2	1/2
II		$P_1 \to (2,1)$ $P_2 \to (1,1)$ $P_3 \to (1,2)$	2	1/2
III		$P_1 \to (1,0)(1,1)$ $P_2 \to (1,0)(1,2)$ $P_3 \to (1,1)$ $P_4 \to (1,2)$	2	1/2
IV		$P_1 \to (1,0)(1,0)(1,1)$ $P_2 \to (1,0)(1,0)(1,2)$ $P_3 \to (1,0)(1,0)(1,3)$ $P_4 \to (1,0)(1,1)$ $P_5 \to (1,0)(1,2)$ $P_6 \to (1,0)(1,3)$ $P_7 \to (1,1)$ $P_8 \to (1,2)$ $P_9 \to (1,3)$	3	1/3
ITAKURA		NO PRODUCTION RULE CHARACTERIZATION	2	1/2

Fig. 5. A summary of the types of local constraints investigated in this paper.

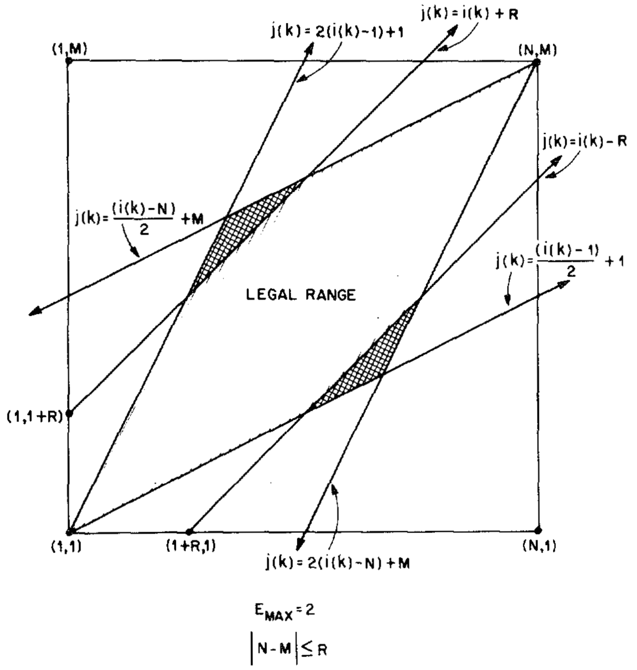


Fig. 6. The effects of global range constraints, and range limiting, on the grid of possible paths for time warping.

For the Type I local constraints of Fig. 3, $E_{MAX} = 2$ and $E_{MIN} = \frac{1}{2}$.

Fig. 5 illustrates four additional types of local constraints that were investigated for isolated word recognition. A pictorial representation of the local paths is given, along with the set of productions, and the values of E_{MAX} and E_{MIN} for each type. Type II local constraints have the same initial and final points as Type I constraints; however, these paths do not go through the intermediate points. Type III local constraints are a generalized form of those proposed by Itakura [4]. Both Type II and Type III constraints have $E_{MAX} = 2$ and $E_{MIN} = \frac{1}{2}$. Type IV constraints are an expanded version of Type III constraints in which the maximum expansion is increased to three. This set was included to see if increases in E_{MAX} would help or hurt the performance of the DTW algorithm. The last type of constraints, denoted as Type Itakura in Fig. 5 is the exact set proposed by Itakura [4]. The crossed out path denotes the nonlinear weighting used by Itakura to prevent a path from staying flat for two consecutive frames. This modification, however, prevents us from characterizing the local constraints by a set of productions. The difference between Itakura's local constraints and Type III local constraints is a subtle one. It is related to the look-ahead (or equivalently the looking back) capability of the local constraints in finding the best path to the point (n, m) in the grid. For Itakura's constraints, the look ahead capability is only one frame. Hence, whenever a best path to the grid point $(n-1, m)$ came from the point $(n-2, m)$ (i.e., it was a flat path), then no best path can ever go from the point $(n-1, m)$ to the point (n, m), i.e., this arc is completely eliminated from all subsequent best paths. However, for Type III constraints, the look-ahead is two levels; hence, even if the best path to the grid point $(n-1, m)$ came from the intermediate point $(n-2, m)$, there can be paths to the grid point (n, m) which pass through the intermediate point $(n-1, m)$ as long as they started from either the points $(n-1, m-1)$ or $(n-2, m-1)$. Thus, for Type III constraints, the true best path can be found even when two horizontally adjacent points both have horizontal arcs as the last segments on their locally optimal paths.

C. Global Path Constraints

Because of the local path constraints, certain parts of the (n, m) plane are excluded from the region in which the optimal warping path can lie. A simple set of relations can be obtained for expressing the boundaries of the allowable regions of the (n, m) plane. These relations are of the form (assuming $E_{MIN} = 1/E_{MAX}$)

$$1 + \frac{(i(k)-1)}{E_{MAX}} \leqslant j(k) \leqslant 1 + E_{MAX}(i(k)-1) \tag{12a}$$

$$M + E_{MAX}(i(k)-N) \leqslant j(k) \leqslant M + \frac{(i(k)-N)}{E_{MAX}}. \tag{12b}$$

Equation (12a) can be interpreted as limiting the range to those grid points which can be reached via a legal path from the point (1, 1), whereas (12b) represents those points which have a legal path to the point (N, M).

Fig. 6 illustrates the effects of the constraints of (12) in the (n, m) plane for a maximum expansion of $E_{MAX} = 2$ (recall that $E_{MIN} = 1/E_{MAX}$). The range of valid paths is restricted to be within the parallelogram (shown by diagonal lines) defined by the lines of slope 2 to 1 and $\frac{1}{2}$ to 1 [6], [4].

An additional restriction on the global range, proposed by Sakoe and Chiba [6], is that

$$|i(k) - j(k)| \leqslant R \tag{13}$$

where R is the maximum allowable absolute time difference (in frames) between test and reference patterns. The effect of this additional restriction is to reduce the size of the parallelogram grid by cutting off the corners, as shown in Fig. 6. The cross-hatched area, in this figure, is the difference between the full parallelogram range [without the restriction of (13)] and the restricted range.

D. Axis Orientation

In (3), we arbitrarily assigned n to $i(k)$ and m to $j(k)$. Another equally arbitrary assignment of variables would be

$$n = j(k), \quad k = 1, 2, \cdots, K \tag{14a}$$

$$m = i(k), \quad k = 1, 2, \cdots, K. \tag{14b}$$

In cases when *both* the local constraints are symmetric, *and* the distance metric is symmetric, there are no differences between the variable assignments of (3) and (14). However, when there is asymmetry in either local constraints (e.g., Type III constraints), or in the distance metric, then the differences in variable assignments of (3) and (14) can be significant. To distinguish these assignments, we refer to those of (3) as "reference along the x-axis" and those of (14) as "test along the x-axis." (Unless otherwise noted, the assignments of (3) will be used in subsequent discussions.)

E. Distance Measure

The last factor of the DTW algorithm is the distance function used to obtain the optimal warping path. A general form for such a distance function is

$$D(i(k), j(k)) = \frac{\sum_{k=1}^{K} d(i(k), j(k))\, \tilde{W}(k)}{N(\tilde{W})} \tag{15}$$

where $D(i(k), j(k))$ is a functional (i.e., a function of a set of functions) that gives the *total* distance along the path of length K, (defined by the K pairs $(i(k), j(k), k = 1, 2, \cdots, K)$, $d(i(k), j(k))$ is the local distance between frames $i(k)$ of the reference, and $j(k)$ of the test, $\tilde{W}(k)$ is a weighting function of the kth arc of the path, and $N(\tilde{W})$ is a normalization factor which is a function of the weighting function $\tilde{W}$.

The definition of the optimal path can be made directly from (15), as the path that minimizes the total distance $D(i(k), j(k))$. More formally, if we denote the minimum path distance (i.e., the distance along the optimal path) as $\hat{D}$, then

$$\hat{D} = \min_{(K, i(k), j(k))} (D(i(k), j(k))). \tag{16}$$

In order to implement the computation of (15) and (16), three functions must be specified, namely, the local distance function d, the weighting function $\tilde{W}$, and the normalization factor $N(\tilde{W})$. The local distance function d depends only on the feature set used to create the test and reference patterns, and is independent of the details of the DTW algorithm. As such, we defer specification of d until Section III of this paper.

The weighting function $\tilde{W}$ depends only on the local path. Four types of weighting functions have been proposed [6]. These have the form

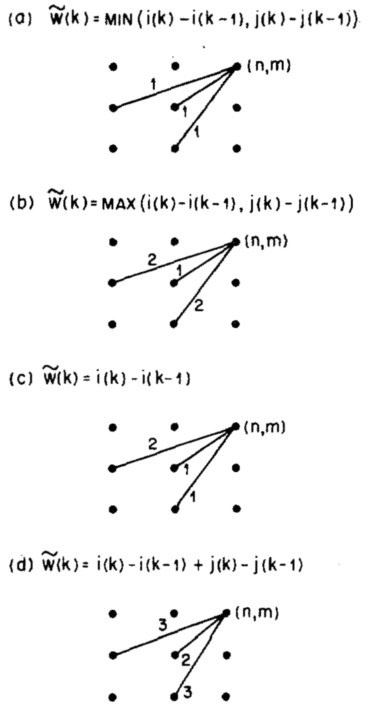

(a) W̃(k) = MIN(i(k) - i(k-1), j(k) - j(k-1))
(n,m)
1
1
1
(b) W̃(k) = MAX(i(k) - i(k-1), j(k) - j(k-1))
(n,m)
2
1
2
(c) W̃(k) = i(k) - i(k-1)
(n,m)
2
1
1
(d) W̃(k) = i(k) - i(k-1) + j(k) - j(k-1)
(n,m)
3
2
3

Fig. 7. Examples illustrating the application of weighting functions to Type II constraints.

$$\tilde{W}(k) = \min (i(k) - i(k-1), j(k) - j(k-1)) \quad \text{(Type a)} \tag{17a}$$

$$\tilde{W}(k) = \max (i(k) - i(k-1), j(k) - j(k-1)) \quad \text{(Type b)} \tag{17b}$$

$$\tilde{W}(k) = i(k) - i(k-1) \quad \text{(Type c)} \tag{17c}$$

$$\tilde{W}(k) = i(k) - i(k-1) + j(k) - j(k-1) \quad \text{(Type d)} \tag{17d}$$

where $i(0)$ and $j(0)$ are defined to be 0 for initialization purposes.

Fig. 7 gives a pictorial representation of the four types of weights as applied to the Type II paths. The number above each arc is the weight attached to that arc. We observe that weighting function Type a weights all arcs (whose slopes are not 0 or ∞) equally, weighting function Type b weights the arcs of slope $\frac{1}{2}$ and 2 more strongly than the arc of slope 1, weighting function Type c weights the arcs according to the distance moved in the x direction, and weighting function Type d weights the arcs according to the sum of the distances moved in the x and y directions. As another example of the use of the weighting function, the left-hand side of Fig. 8 shows the weights applied to Type I paths. It can be seen that, for Types a and c weights, some arcs of the path receive 0 weight. For such cases the local distance does not contribute to the overall distance. To eliminate such a nonphysical occurrence, the use of a smoothing function on the weights was suggested by Sakoe and Chiba [6]. The smoothing function averages the weights along multiple segments of a local path. The result of applying the smoothing to the weights of Type I paths is shown on the right side of Fig. 8. In this manner equal weight is given to all segments of a local path.

The choice of the normalization factor $N(\tilde{W})$ is determined by the constraint that the total distance $D(i(k), j(k))$ be the

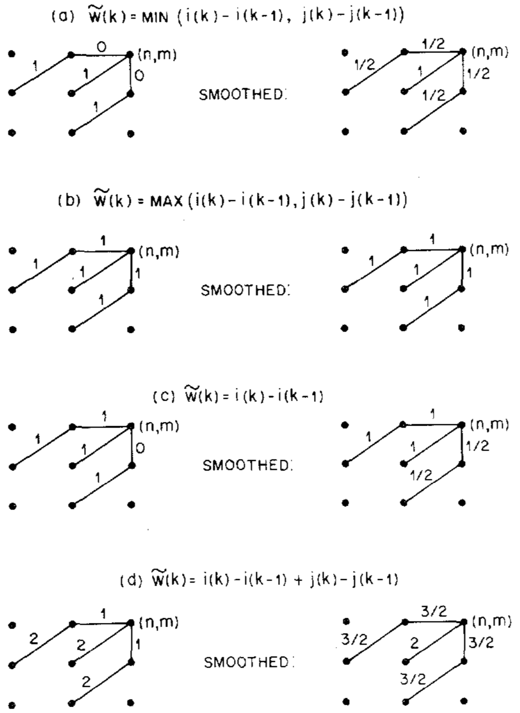

(a) $\widetilde{W}(k) = \text{MIN}\,(i(k) - i(k-1),\ j(k) - j(k-1))$
0
(n,m)
1
1
0
1
SMOOTHED:
1/2
(n,m)
1/2
1
1/2
1/2
(b) $\widetilde{W}(k) = \text{MAX}\,(i(k) - i(k-1), j(k) - j(k-1))$
1
(n,m)
1
1
1
1
SMOOTHED:
1
(n,m)
1
1
1
1
(c) $\widetilde{W}(k) = i(k) - i(k-1)$
1
(n,m)
1
1
0
1
SMOOTHED:
1
(n,m)
1
1
1/2
1/2
(d) $\widetilde{W}(k) = i(k) - i(k-1) + j(k) - j(k-1)$
1
(n,m)
2
2
1
2
SMOOTHED:
3/2
(n,m)
3/2
2
3/2
3/2

Fig. 8. Examples illustrating the effects of smoothed weighting functions on the paths of Type I constraints.

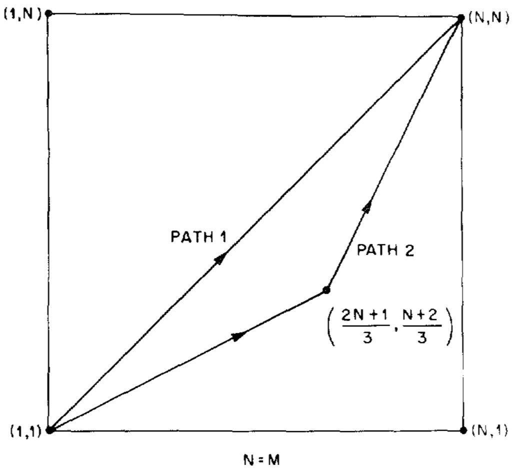


Fig. 9. Illustration of two possible paths through a grid.

average local distance along the path and, as such, is independent of both the lengths of the reference and test patterns, and also is independent of the length of the particular time alignment path. As such, the normalization is of the form

$$N(\widetilde{W}) = \sum_{k=1}^{K} \widetilde{W}(k). \tag{18}$$

For Types c and d weighting functions, the normalization is

$$N(\widetilde{W}_c) = \sum_{k=1}^{K} (i(k) - i(k-1)) = i(K) - i(0) = N \tag{19a}$$

$$N(\widetilde{W}_d) = \sum_{k=1}^{K} (i(k) - i(k-1) + j(k) - j(k-1))$$

$$= i(K) - i(0) + j(K) - j(0) = N + M. \tag{19b}$$

However, for Types a and b weighting functions, $N(\widetilde{W})$, as defined in (18), depends strongly on the path. This result is illustrated in Fig. 9 which shows two paths in a grid ($N = M$ for convenience). Path 1 is a straight line of slope 1 from (1, 1) to (N, N). Path 2 has two sections, the first of which has slope $\frac{1}{2}$ (from (1, 1) to $((2N+1)/3, (N+2)/3)$) and the second has slope 2 (from $((2N+1)/3, (N+2)/3)$ to (N, N)). For Type a weights, the normalization along the path is

$$\text{Path 1:}\quad N(\widetilde{W}_a) = N \tag{20a}$$

$$\text{Path 2:}\quad N(\widetilde{W}_a) = 2N/3 \tag{20b}$$

Similarly, for Type b weights we get

$$\text{Path 1:}\quad N(\widetilde{W}_b) = N \tag{21a}$$

$$\text{Path 2:}\quad N(\widetilde{W}_b) = 4N/3. \tag{21b}$$

While it is possible, in principle, to compute normalization factors of the form of (18), it is not possible when the computation is performed by a recursive (dynamic programming) algorithm, since in such an algorithm the minimization process is performed locally, and hence a path dependent normalization of this type is clearly inappropriate. As such, an arbitrary normalization must be chosen for Types a and b weights. For these cases we define the normalization to be

$$N(\widetilde{W}_a) = N \tag{22}$$

and similarly for Type b weights we use

$$N(\widetilde{W}_b) = N. \tag{23}$$

Clearly, the normalizations of (22) and (23) for Types a and b weights leads to a bias in the DTW algorithm, i.e., a preference for some paths over others. For Type a weighting, the DTW algorithm has a preference for longer paths, whereas for Type b weighting, the preference is for shorter paths. It is anticipated that this bias will affect the performance of the DTW algorithm, and may prevent the method from finding the optimal path.[2]

F. Dynamic Time Warping Implementation

In order to implement a time warping algorithm in a dynamic programming manner, two basic principles are used, namely:

1) a globally optimal path is also locally optimal;

2) the optimal path to the grid point (n, m) only depends on values of n', m' such that $n' \leqslant n$, $m' \leqslant m$.

These two principles define the standard dynamic programming recursive relationship. Using them, it is possible to create a partial accumulated distance function $D_A(n, m)$, representing the accumulated distance along the best path from (1, 1) to (n, m), of the form

$$D_A(n, m) = \min_{\substack{(i(k), j(k), K') \\ \text{s.t. } i(K') = n,\, j(K') = m}} \left[\sum_{k=1}^{K'} d(i(k), j(k))\, \widetilde{W}(k) \right]. \tag{24}$$

[2] It should be noted that White and Neely [2] used Type b weights. This may account for their lack of improvement of the DTW time alignment over simple linear time alignment for the alpha-digit vocabulary.

$D_A(n, m)$ depends only on the paths from $(1, 1)$ to (n, m) and can be defined recursively in terms of an intermediate point (n', m') where $n' < n$, $m' < m$, as

$$D_A(n, m) = \min_{(n', m')} [D_A(n', m') + \hat{d}((n', m'), (n, m))] \quad (25)$$

where $\hat{d}$ is the weighted distance from $(n'_., m)$ to (n, m), i.e.,

$$\hat{d}((n', m'), (n, m)) = \sum_{l=0}^{L-1} d(i(K' - l), j(K' - l))\, \tilde{W}(K' - l), \quad (26)$$

and in which L is the number of segments in the path from (n', m') to (n, m), and where

$$i(K') = n, \qquad j(K') = m \quad (27a)$$

$$i(K' - L) = n', \quad j(K' - L) = m'. \quad (27b)$$

For an efficient implementation of (25), it is only necessary to restrict the range of (n', m') of (25) to the set of grid points which use a single production to reach (n, m) from (n', m'). Fig. 10 illustrates this important point with four examples. The first example uses Type I local constraints and Type a weights (smoothed) and is shown in part a of Fig. 10. In this case (25) becomes

$$D_A(n, m) = \min \begin{bmatrix} D_A(n - 1, m - 1) + d(n, m) \\ D_A(n - 1, m - 2) + \frac{1}{2}(d(n, m - 1) + d(n, m)) \\ D_A(n - 2, m - 1) + \frac{1}{2}(d(n - 1, m) + d(n, m)) \end{bmatrix}. \quad (28)$$

Similarly, part b shows an example of Type II constraints with Type d weighting, and part c shows an example of Type III constraints with Type c weighting. In part d we show Itakura's DTW algorithm. The purpose of the $g(k)$ function is to disallow paths which go horizontally for more than one frame.

Whenever $N(\tilde{W})$ is independent of the path, or equivalently, whenever a solution to the unnormalized minimization problem provides a solution to the normalized minimization problem, it is possible to minimize (15) as follows:

$$\hat{D} = \min_{(i(k), j(k), K)} \left[\frac{\sum_{k=1}^{K} d(i(k), j(k))\, \tilde{W}(k)}{N(\tilde{W})} \right] \quad (29a)$$

$$= \frac{\min_{(i(k), j(k), K)} \left[\sum_{k=1}^{K} d(i(k), j(k))\, \tilde{W}(k) \right]}{N(\tilde{W})}. \quad (29b)$$

In such cases we may use (25) to give

$$\hat{D} = \frac{D_A(N, M)}{N(\tilde{W})} \quad (30)$$

and the implementation of the DTW algorithm is a three step procedure.

1) Initialization: Set $D_A(1, 1) = d(1, 1)\, \tilde{W}(1)$.

2) Recursion: Compute $D_A(n, m)$ recursively for $1 \leqslant n \leqslant N$, $1 \leqslant m \leqslant M$.

3) Termination: Set $\hat{D} = D_A(N, M)/N(\tilde{W})$.

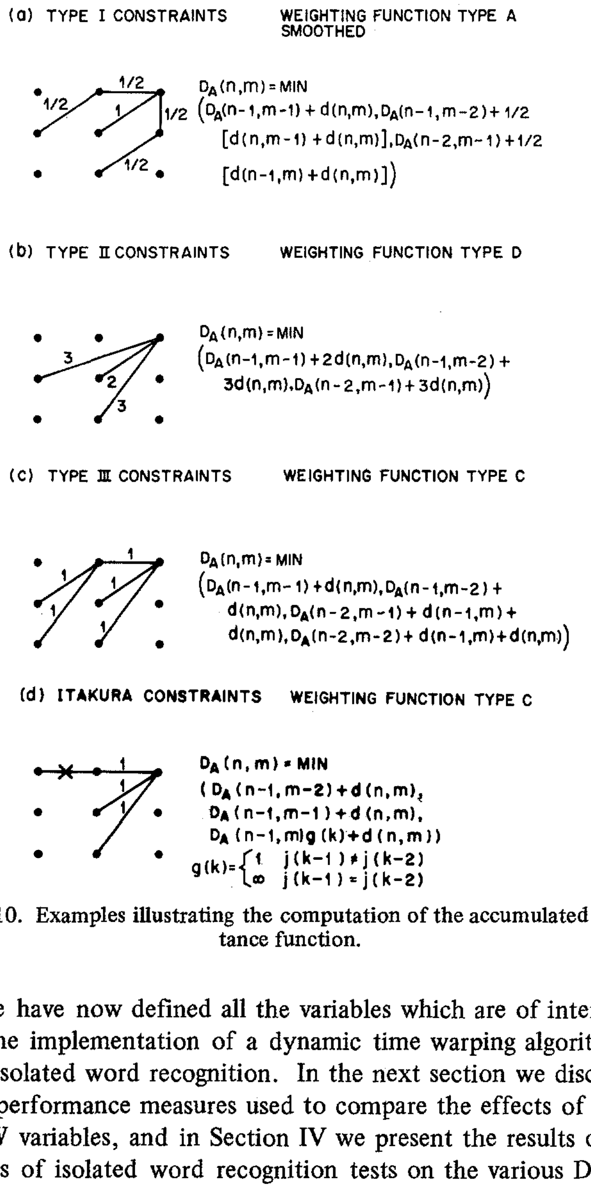

(a) TYPE I CONSTRAINTS
WEIGHTING FUNCTION TYPE A SMOOTHED
1/2
1/2
1
1/2
1/2
$D_A(n,m) = \mathrm{MIN}$
$(D_A(n-1,m-1) + d(n,m), D_A(n-1,m-2) + 1/2$
$[d(n,m-1) + d(n,m)], D_A(n-2,m-1) + 1/2$
$[d(n-1,m) + d(n,m)])$
(b) TYPE II CONSTRAINTS
WEIGHTING FUNCTION TYPE D
3
2
3
$D_A(n,m) = \mathrm{MIN}$
$(D_A(n-1,m-1) + 2d(n,m), D_A(n-1,m-2) +$
$3d(n,m), D_A(n-2,m-1) + 3d(n,m))$
(c) TYPE III CONSTRAINTS
WEIGHTING FUNCTION TYPE C
1
1
1
1
1
$D_A(n,m) = \mathrm{MIN}$
$(D_A(n-1,m-1) + d(n,m), D_A(n-1,m-2) +$
$d(n,m), D_A(n-2,m-1) + d(n-1,m) +$
$d(n,m), D_A(n-2,m-2) + d(n-1,m) + d(n,m))$
(d) ITAKURA CONSTRAINTS
WEIGHTING FUNCTION TYPE C
1
1
1
$D_A(n, m) = \mathrm{MIN}$
$(D_A(n-1, m-2) + d(n, m),$
$D_A(n-1, m-1) + d(n, m),$
$D_A(n-1, m) g(k) + d(n, m))$
$g(k) = \begin{cases} 1 & j(k-1) \neq j(k-2) \\ \infty & j(k-1) = j(k-2) \end{cases}$

Fig. 10. Examples illustrating the computation of the accumulated distance function.

We have now defined all the variables which are of interest in the implementation of a dynamic time warping algorithm for isolated word recognition. In the next section we discuss the performance measures used to compare the effects of the DTW variables, and in Section IV we present the results of a series of isolated word recognition tests on the various DTW algorithms.

III. Performance Criteria and Measurements for DTW Algorithms

Although the DTW algorithms of Section II can be evaluated in a variety of applications, we are only concerned here with their performance in an isolated word recognition system of the type shown in Fig. 11 [4], [5]. This recognition system has been applied in a variety of tasks [4], [5], [9]-[12] and has been found to be a reliable and robust system for isolated word recognition. The analysis features are a set of $(p + 1)$ $(p = 8)$ autocorrelation coefficients for each frame, where a frame is 45 ms of speech, and frames overlap by 30 ms (i.e., 67 frames/s). The local distance for comparing frames of the reference and test is the log likelihood ratio distance proposed by Itakura [4], [13]. The output of the DTW algorithm for the vth reference word is the distance $\hat{D}^{(v)}$ (30), and the decision rule processes the set of $\hat{D}^{(v)}$, scores to determine an ordered list of recognition candidates. The "recognized" word is generally the candidate with the smallest value of $\hat{D}^{(v)}$.

In a practical implementation of the recognition system of Fig. 11, it is found that the DTW computation contributes

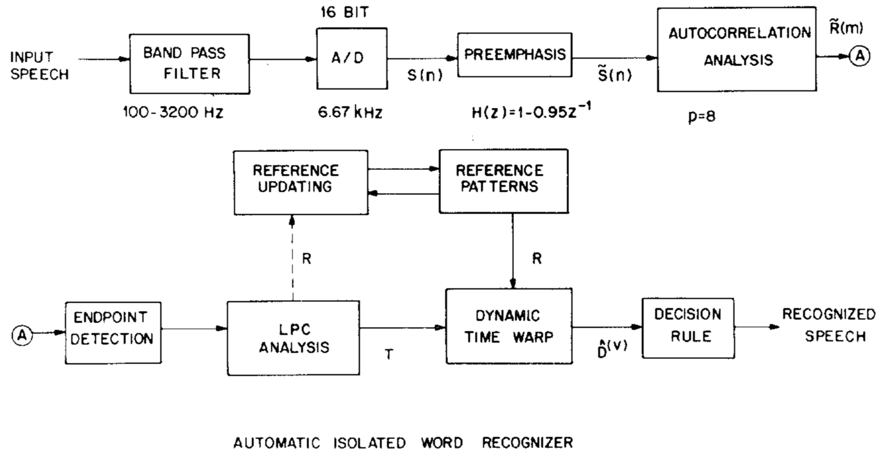


Fig. 11. A block diagram of the isolated word recognizer used in the performance evaluations.

substantially to the overall computation of the system (typically from 50 to 90+ percent, depending on the number of reference patterns). As such, speed of computation is an important consideration in evaluating the performance of a DTW algorithm. Similarly, the required space (memory) for implementing the algorithm is an important factor since we would like the overall recognition system to be easily implemented in either general purpose software, or special purpose hardware. Typically, a DTW algorithm is computed as a series of vectors, i.e., n is fixed and $D_A(n, m)$ is computed for all m. As such, the natural unit to measure storage is the number and size of the vectors needed. In most applications vectors are stored for both $D_A(n, m)$ and $d(n, m)$. Finally, the last, and most important, index of performance is the accuracy with which the DTW algorithm finds the best path for the spoken word, as measured by the word recognition accuracy of the system.

Thus, the three performance criteria of the DTW algorithms are as follows.

1) *Memory Requirements:* The number and size of the vectors that need to be stored in order to compute the accumulated distance function $D_A(n, m)$.

2) *Efficiency (Speed of Computation):* The amount of time required by the time warping algorithm to compute the optimal path. For this measure there are two components, namely, time for combinatorics and time for local distance calculations. The combinatorics time is the time to compute the accumulated distance function, given the values of the local distance function.

3) *Recognition Accuracy:* The probability (measured as the percentage of occurrences) that the reference word with the smallest distance $\hat{D}^{(v)}$ matches the spoken word in a series of isolated word recognition tests.

In the next section we study the effects of varying the parameters of the DTW algorithms on each of the performance criteria.

IV. Results of Performance Evaluations

A series of recognition tests were performed using the recognizer of Fig. 11. Two test sets of data were used.

1) TS1–Two talkers, (SD1 and SD2), speaker dependent reference patterns (2/word), 39 word vocabulary (A-Z, 0-9, STOP, ERROR, REPEAT), five replications of each word for each talker in the test set.

2) TS2–Four talkers (SI1, SI2, SI3, and SI4), speaker independent reference patterns (2/word) generated from a clustering analysis [11], 54 word vocabulary of computer terms [11], one replication of each word for each talker.

All recorded words were obtained from previous investigations of word recognition [5], [11]. A total of 606 words (5 X 39 X 2 + 4 X 54) were used in each recognition test.

The entire experimental system was implemented in Fortran on a Data General Eclipse S230 minicomputer. Measurements of memory usage were made from the Fortran code of the DTW algorithms. Measurements of computational speed were made by averaging the results of 1500 separate time warps using a computer controlled microsecond clock that was accurate to ±10 μs. Recognition accuracy scores were obtained by counting the number of correct recognitions and normalizing by the number of spoken words.

A. Memory Requirements

As mentioned previously, the differences among the DTW algorithms with respect to storage are related to the number and the size of the vectors needed for accumulated distances, local distances, and possibly side information (for Itakura's local constraints). Table I summarizes the requirements on the number of vectors (all of size M) needed for each of the types of local constraints. (No other DTW parameter affects memory size.) Both Type II and Itakura constraints each require two vectors, while Types I and III require three vectors, and Type IV constraints require five vectors. (It should be noted that one of the two vectors for Itakura's constraints is the side information–the g function–which can be coded to 1 bit.) Since the total storage required by any of the constraint types is small (M typically is less than 75 for isolated words), it is concluded that memory requirements are not a factor in the choice of a DTW algorithm.

B. Computational Efficiency

The results on computational speed are presented in Table II and Fig. 12. Table II-A shows the effect of the type of local constraint on the average time required to perform the combinatorics part of the DTW algorithm. Both Type II and

TABLE I
THE MEMORY REQUIREMENTS FOR DTW ALGORITHMS AS A FUNCTION OF THE TYPE OF LOCAL CONSTRAINTS

	Local Constraints				
Vectors Needed	I	II	III	IV	Itakura
Accumulated Distance	2	2	2	3	1
Local Distance	1	0	1	2	0
Side Information	0	0	0	0	1
Total	3	2	3	5	2

TABLE II
COMPARISONS OF TIMINGS AMONG THE DTW ALGORITHMS

	Local Constraints				
Orientation	I	II	III	IV	Itakura
Reference Along x-axis	85.1	63.2	82.8	249.3	63.2
Test Along x-axis	86.5	63.6	83.0	-	63.7
Average	85.8	63.4	82.9	249.3	63.5

Average Combinatorics Time (Milliseconds) Per Warp
(A)

	Weighting Function			
Local Constraints	Type a	Type b	Type c	Type d
I	90.2	80.6	85.1	90.8
II	57.8	65.9	63.2	69.6
Average	74.0	73.3	74.2	80.2

Average Combinatorics Time (Milliseconds) Per Warp
(B)

Local Constraint	Average Number of Distance Calculations
I	543.2
II	491.7
III	504.4
IV	781.0
Itakura	504.4

Average Local Distance Calculations Per Warp
(C)

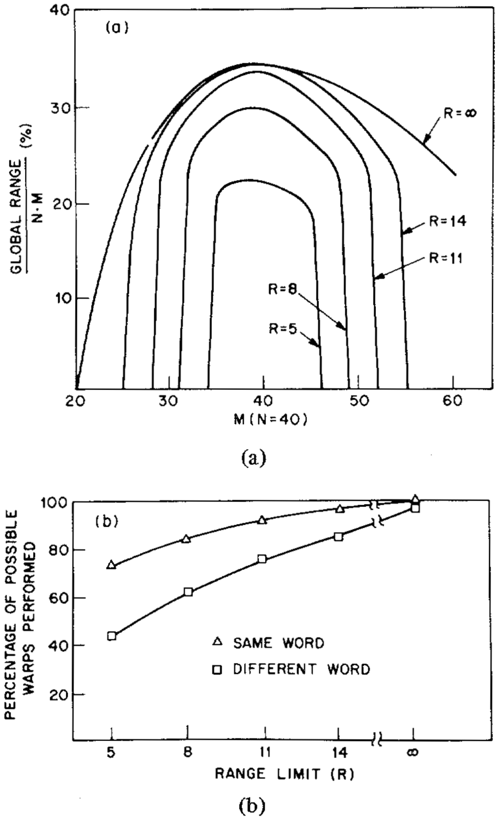
Itakura constraints required about 25 percent less time than Type I and Type III constraints. The time for Type IV constraints (where E_{MAX} was 3) was about 3-4 times greater than the time for any other type of local constraint. Table II-A also shows that the orientation of the test and reference patterns (with respect to the x-axis) was irrelevant as far as the timing for conbinatorics was concerned.

Table II-B shows that the weighting function had only a negligible effect on the time for combinatorics for both Type I and Type II local constraints.

The major factor which affected computational efficiency was the number of local distance calculations (a dot product on vectors of length $p + 1$) that had to be performed for each point within the range of a given time warp. Table II-C shows the effect of the local constraints on the average number of local distance calculations. Types II, III, and Itakura all had the smallest average number of local distance calculations, with Type I constraints requiring only a somewhat larger number. Type IV constraints required a 50 percent increase in the average number of local distance computations—a significant overall increase in computation time.

Since local distance calculations required about 80 percent of the total computation time for the DTW algorithm, techniques for reducing the number of such calculations are potentially very useful. The range limiting technique of Section II (as proposed by Sakoe and Chiba [6]) is one such technique. Fig. 12(a) illustrates the reduction in global range (i.e., number of distance calculations) as a function of M (for $N = 40$) and R. It can be seen that as R goes from ∞ (no range limiting) to $R = 5$, a reduction of about 50 percent in the global range occurs when $M = N = 40$. It can also be seen in Fig. 12(a) that when $|N - M|$ approaches R, a very sudden reduction in the global range occurs, since the endpoints of a path (N, M) no longer are points within the global range, i.e., there is no legal path from $(1, 1)$ to (N, M).

Fig. 12(b) shows another effect of range limiting. Only those reference-test pairs whose lengths are sufficiently close, i.e., $|N - M| \leq R$, can be compared via a fixed range DTW algorithm. Here we plot the percentage of possible warps which are actually performed. Two cases are shown, namely, the first, in which the reference and test pattern represent the same word, and the second, in which the reference and test pattern represent different words. It would be ideal if no reductions occurred when the test and reference were the same word; however, this is not the case, as seen in Fig. 12(b). Although the reduction in range for "same" words is considerably smaller than for "different" words, a fairly strong effect is noted for "same." We will discuss this issue in more detail in Section IV where we show how range limiting can be successfully applied in a normalize/warp DTW algorithm.



(a)

(b)

Fig. 12. Plots showing the effects of range limiting on (a) the percentage of the total range used and (b) the percentage of possible warps that are performed.

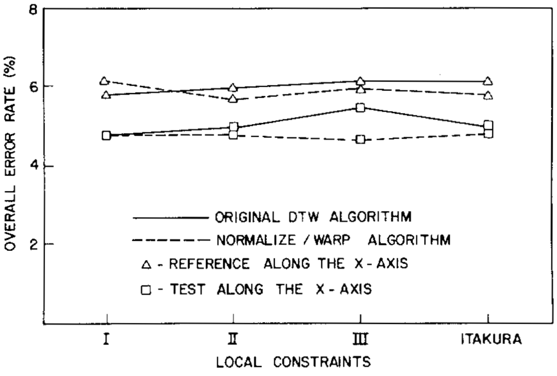
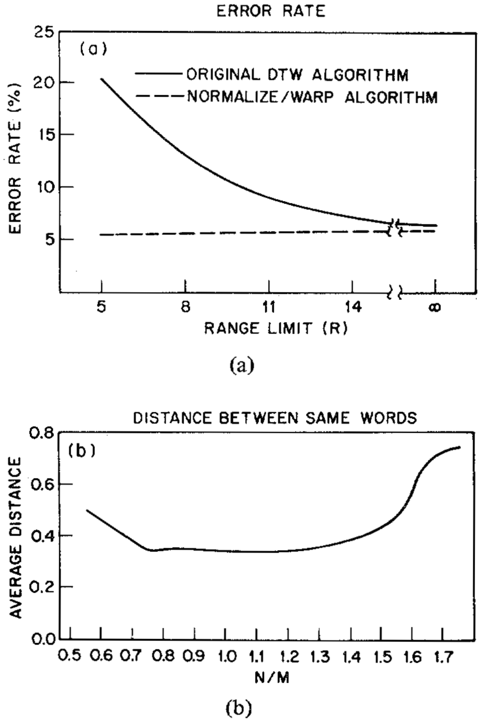


Fig. 13. Plots showing the effects of local constraints, and axis orientation, on overall error rate for both the standard DTW algorithm (solid lines) and the normalize/warp DTW algorithm (dashed lines).

C. Recognition Accuracy

The results on recognition accuracy are presented in Figs. 13 and 14 and Table III. The solid curves of Fig. 13 show the average recognition error rate as a function of the type of local constraints for both reference and test along the x-axis. (The dashed lines will be explained in Section V.) Weighting function Type c with no range limitations was used in obtaining all the recognition scores. It can be seen that the local constraints have negligible effect on the error rate.[3] However, the axis orientation shows a consistent decrease in error rate when the test is along the x-axis rather than the reference. This effect has been previously noted [7].



(a)

(b)

Fig. 14. Plots showing (a) the effects of ranging limiting on error rate for both the standard DTW algorithm (solid line) and the normalize/warp DTW algorithm (dashed line) and (b) the effect of the ratio N/M on the average distance between reference and test when both reference and test were the same words.

In order to understand why axis orientation is important we must examine the effects of the choice of weighting function on the performance of the various DTW algorithms. In Table III we show the total number of errors for local constraints I and II as a function of the choice of weighting function. The results for weighting functions Types a, b, c, and d were computed with reference along the x-axis. Weighting function Type c$'$ is weighting function Type c as computed with test along the x-axis. Weighting function Type c is used twice because it is the only asymmetric weighting function. We observe that, while weighting function Type c is the worst for reference along the x-axis, it is the best overall when computed with test along the x-axis. Thus, we must conclude that improvements in recognition accuracy that occur when the test pattern is along the x-axis are due to some property of weighting function Type c. As Sakoe and Chiba observed, the use of weighting function Type c is equivalent to integration along the x-axis [6]. Based on the above reasoning, we conclude that by applying an equal weight to all test frames, a better differentiation between "same" and "different" pairs is achieved.

Examination of Table III reveals another interesting result regarding the performance of a DTW algorithm as a function of the choice of weighting function. We observe that, in agreement with the results previously reported by Sakoe and Chiba [6], a symmetric weighting function (weighting function Type d) performs better than an asymmetric weighting function (weighting function Type c) when the reference pattern is placed along the x-axis, but that biased weighting functions

[3] Type IV local constraints, however, gave significantly higher error values than the other local constraints, and hence are not given in Fig. 13. This effect has been noted previously [6].

TABLE III
TOTAL NUMBER OF ERRORS AS A FUNCTION OF THE TYPE OF WEIGHTING FUNCTION

	Weighting Function				
Speaker	Type a	Type b	Type c	Type d	Type c′
SD1	22	21	21	22	23
SD2	8	7	8	9	7
SI1	11	8	10	10	10
SI2	7	12	14	10	7
SI3	12	13	14	12	10
SI4	5	1	4	4	2
Total	65	62	71	67	59

Total Errors

Local Constraints Types I and II

(weighting functions Types a and b) perform better, not worse, than unbiased weighting functions. However, since the largest difference in error rate between weighting functions is small, and since the relative error rates are not constant over all speakers, we conclude that there is no significant difference in the performance of a DTW algorithm regarding the choice of weighting function, but that the combination of test along the x-axis and weighting function Type c provides significant improvement in recognition accuracy.

The effects of range limiting on the error rate are shown by the solid curve in Fig. 14(a). This curve (obtained using Type III local constraints, Type c weighting, and reference along the x-axis) shows that as R decreases, the error rate of the system rapidly increases. Thus, we conclude that too small a value of R (too small a range) is harmful to the performance of the DTW algorithm. This point is illustrated in Fig. 14(b) which shows a plot of the average DTW distance (when reference and test are the same words) as a function of N/M, the ratio of the length of the reference to the length of the test. It is seen that as N/M approaches $\frac{1}{2}$ or 2, the average distance increases rapidly, indicating that there is very little area in the (n, m) plane to find a good warping path. However, near $N/M = 1$ the average distance is essentially constant. We will use these observations in Section V to propose a modified DTW algorithm in which range limiting is generally helpful to the performance of the algorithm.

V. Discussion of Results

The results presented in the previous section show the following.

1) There is little performance difference among the local constraints of Types I, II, III and Itakura; however, local constraint Type IV has significantly worse performance than the other types.

2) Range limiting serves to reduce the computation time but to increase the error rate of the system. At what point this tradeoff should be made depends heavily on the cost of increased computation versus the cost of increased error rate. However, we will show later that it is possible to reduce computation without loss of accuracy.

3) Small but consistent improvements in recognition accuracy were obtained when the test pattern was oriented along the x-axis.

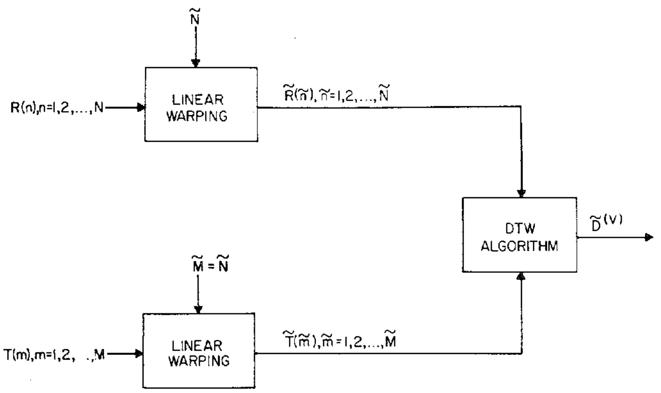


Fig. 15. Processing used in the normalize/warp DTW algorithm.

4) Weighting function Type c provided the best recognition scores for test along the x-axis, whereas weighting function Type b provided the best recognition scores for reference along the x-axis. The differences in error rate, among the different weighting functions, however, were small and hence these differences may not be significant.

5) The DTW algorithms performed best when the ratio of the length of reference to the length of test approached 1, and worst when the ratio approached $\frac{1}{2}$ or 2.

The analysis of the above results led to a modified DTW algorithm, called the normalize/warp method, in which a length normalizing preprocessor is used on both the reference and test patterns, prior to the DTW algorithm. Such an algorithm has been in use at Threshold Technology since 1978, and has recently been described by Welch [14]. The form of the processing is illustrated in Fig. 15. The normalizing preprocessor linearly interpolates (or decimates) the length of the reference and test patterns to a fixed length $\tilde{N}$ so that the resulting length ratio $\tilde{N}/\tilde{M}$ is 1. The normalized reference pattern $\tilde{R}(\tilde{n})$ is given as

$$\tilde{R}(\tilde{n}) = (1 - s)R(n) + s(R(n + 1)), \qquad \tilde{n} = 1, 2, \cdots, \tilde{N} \quad (31)$$

where

$$n = \left\lfloor (\tilde{n} - 1)\frac{(N - 1)}{(\tilde{N} - 1)} + 1 \right\rfloor \quad (32a)$$

$$s = (\tilde{n} - 1)\frac{(N - 1)}{(\tilde{N} - 1)} + 1 - n \quad (32b)$$

and $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x. Similarly, the test pattern $T(m)$ is linearly warped to give $\tilde{T}(\tilde{m})$. It can be shown that linear interpolation (or decimation) is adequate for the patterns we are working with by examining the log spectrum of any component of a reference or a test pattern. It has been shown [15], that the log spectra of these signals are extremely band limited; hence simple linear warping is adequate. The value of $\tilde{N}$ used in this system was 40, which was the average length over all words that were used in the test.

Following length normalization, a DTW algorithm was applied to the normalized patterns. Both the memory requirements and the computational efficiency of the modified DTW algorithm are, on average, the same as for the length unnormalized data. The effects of the normalize/warp algorithm on recognition accuracy are shown by the dashed lines in Figs. 13 and 14(a). From Fig. 13 we see that, in general, the normalize/warp DTW algorithm performs *slightly better* than the unnormalized DTW algorithm, in almost all cases. More importantly, perhaps, Fig. 14(a) shows that range limiting does not increase the error rate of the normalize/warp algorithm; in fact the error rate decreased slightly as R went from ∞ to 5.[4] One reasonable explanation for this behavior is that the normalization of lengths is equivalent to a linear time warping procedure and that the actual nonlinear component is not very large. This result has major practical significance, since range limiting is a useful technique for reducing the computation involved in the DTW algorithm.

The normalize/warp approach to dynamic time warping has significant implementational advantages because of the fixed length of the reference and test patterns. For example, the fixed length reference patterns can be rapidly accessed without an address table. Furthermore, the addressing and range computation in the DTW algorithm can be performed once and stored, thereby reducing overhead in this part of the computation. Finally, since all reference and test patterns are of a fixed length, no normalization of distances is required. Hence, the final decision can be made on the basis of unnormalized distance scores.

VI. Summary

The purpose of this paper was to study the effects of the variable parameters of a DTW algorithm on its performance in an isolated word speech recognition system. It was found that the best performance among conventional DTW algorithms was obtained with Type c weighting, test along the x-axis, any of the simple local continuity constraints (Types I, II, III or Itakura), and no global range limitations. The only real tradeoff in performance was between speed and accuracy when range limiting was applied.

More importantly, however, the performance results led to a modified version of the DTW algorithm called the normalize/warp method whose performance was shown to be as good or better than the other DTW algorithms that were investigated. This new algorithm was also shown to have a number of implementational advantages over conventional DTW algorithms.

Acknowledgment

The authors would like to acknowledge the thoughtful comments and criticisms of the anonymous reviewers.

References

[1] T. B. Martin, "Practical applications of voice input to machines," *Proc. IEEE*, vol. 64, pp. 487-501, Apr. 1976.

[2] G. M. White and R. B. Neely, "Speech recognition experiments

[4] As R is reduced below 5 the error rate increases. We have found that the error rate for $R = 5$ was 5.3 percent, and for $R = 1$ it was 8.9 percent.

with linear prediction, bandpass filtering and dynamic programming," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, pp. 183-188, Apr. 1976.

[3] H. Sakoe and S. Chiba, "A dynamic programming approach to continuous speech recognition," in *Proc. Int. Congr. Acoust.*, Budapest, Hungary, 1971, Paper 20C-13.

[4] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 57-72, Feb. 1975.

[5] 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.

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

[7] 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.

[8] J. E. Hopcroft and J. D. Ullman, *Formal Languages and Their Relation to Automata.* Reading, MA: Addison-Wesley, 1969.

[9] 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.

[10] A. E. Rosenberg and F. Itakura, "Evaluation of an automatic word recognition system over dialed-up telephone lines" (abstract), *J. Acoust. Soc. Amer.*, vol. 60, p. PS12, Nov. 1976.

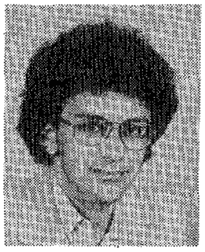
[11] L. R. Rabiner and J. G. Wilpon, "Speaker independent, isolated word recognition for a moderate size (54 word) vocabulary," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 583-587, Dec. 1979.

[12] L. R. Rabiner, J. G. Wilpon, and A. E. Rosenberg, "A voice controlled repertory dialer system," *Bell Syst. Tech. J.*, vol. 59, no. 7, pp. 1153-1163, Sept. 1980.

[13] J. M. Tribolet, L. R. Rabiner, and M. M. Sondhi, "Statistical properties of an LPC distance measure," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 550-558, Oct. 1979.

[14] J. R. Welch, "Combination of linear and nonlinear time normalization for isolated word recognition" (abstract), *J. Acoust. Soc. Amer.*, vol. 67, p. S14, Spring 1980.

[15] C. S. Myers, "A comparative performance study of several dynamic time warping algorithms for speech recognition," M.S. thesis, M.I.T., Cambridge, MA, Feb. 1980.



Cory Myers (S'78) was born on February 24, 1957, in Rochester, NY. He received the B.S. and M.S. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1980.

From 1977 through 1980 he participated in a cooperative program for electrical engineering and computer science at Bell Laboratories, Holmdel, and Murray Hill, NJ, where he worked on computer graphics, digital circuit design, and dynamic programming for speech recognition. He is currently a doctoral student in the Department of Electrical Engineering and Computer Science, M.I.T., and is a Research Assistant in the digital signal processing group. His interests include speech processing and recognition and digital signal processing.

Lawrence R. Rabiner (S'62-M'67-SM'75-F'76), for a photograph and biography, see p. 78 of the February 1980 issue of this TRANSACTIONS.

Aaron E. Rosenberg (S'57-M'63) received the S.B. and S.M. degrees from the Massachusetts Institute of Technology, Cambridge, in 1960, and the Ph.D. degree from the University of Pennsylvania, Philadelphia, in 1964, all in electrical engineering.

Since 1964, he has been with Bell Laboratories, Murray Hill, NJ. He is presently engaged in studies of systems for man-machine communication-by-voice in the Acoustics Research Department at Bell Labs.

Dr. Rosenberg is a member of Eta Kappa Nu, Tau Beta Pi, and Sigma Xi, a Fellow in the Acoustical Society of America, and a member of the IEEE Acoustics, Speech, and Signal Processing Society's Technical Committee on Speech Processing.