A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition

CORY S. MYERS, STUDENT MEMBER, IEEE, AND LAWRENCE R. RABINER, FELLOW, IEEE

Abstract—Dynamic time warping has been shown to be an effective method of handling variations in the time scale of polysyllabic words spoken in isolation. This class of techniques has recently been applied to connected word recognition with high degrees of success. In this paper a level building technique is proposed for optimally time aligning a sequence of connected words with a sequence of isolated word reference patterns. The resulting algorithm, which has been found to be a special case of an algorithm previously described by Bahl and Jelinek, is shown to be significantly more efficient than the one recently proposed by Sakoe for connected word recognition, while maintaining the same accuracy in estimating the best possible matching string. An analysis of the level building method shows that it can be obtained as a modification to the Sakoe method by reversing the order of minimizations in the two-pass technique with some subsequent processing. This level building algorithm has a number of implementation parameters that can be used to control the efficiency of the method, as well as its accuracy. The nature of these parameters is discussed in this paper. In a companion paper we discuss the application of this level building time warping method to a connected digit recognition problem.

Glossary of Terms

$T(m)$	Test pattern.
M	Length (in frames) of test pattern.
$R_v(n)$	Reference pattern v.
V	Number of reference words.
N_v	Length (in frames) of vth reference pattern.
R^s	Super reference pattern consisting of a sequence of concatenated reference patterns.
L	Number of reference patterns in a string.
$\phi(l)$	Length (in frames) of l concatenated reference patterns. (Also the frame of the super reference pattern which corresponds to the end of the lth reference pattern.)
w	Dynamic warping path.
$d(m,n)$	Local distance between mth frame of the test pattern, and the nth frame of the super reference pattern.
D	Global distance between test pattern and super reference pattern.
$D_A(m,n)$	Accumulated distance to frame m of the test pattern and frame n of the super reference pattern.
$L(m)$	Lower boundary function.
$U(m)$	Upper boundary function.
$\tilde{L}(n)$	Lower warping function constraint.
$\tilde{U}(n)$	Upper warping function constraint.
$D_l(m,n)$	Accumulated distance to frame m of the test pattern, and frame n of the lth reference of the super reference pattern.
$\tilde{D}_l(m)$	Accumulated distance to frame m of the test pattern, and the last frame of the lth reference of the super reference pattern.
$d_l(m,n)$	Local distance between the mth frame of the test pattern, and the nth frame of the lth reference of the super reference pattern.
$L_l(m)$	Modified lower boundary function for the lth level.
$U_l(m)$	Modified upper boundary function for the lth level.
e_l	The frame of the test pattern where the optimal path maps e_l to $\phi(l)$ for the lth reference of the super reference pattern.
L_{MIN}	Minimum number of references in a super reference pattern.
L_{MAX}	Maximum number of references in a super reference pattern.
$D_{q(1)q(2)\cdots q(l)}(m)$	Distance between test pattern and super reference pattern $R_{q(1)} \oplus R_{q(2)} \oplus \cdots \oplus R_{q(l)}$ up to frame m of the test pattern.
$\tilde{R}^s_L$	Best super reference pattern of length L that matches the test pattern.
$\tilde{D}^B_l(m)$	Minimum value of $\tilde{D}_l(m)$ over all possible super reference patterns of length l.
$W_l(m)$	The index v, of the reference pattern R_v, that gives $\tilde{D}^B_l(m)$.
$F_l(m,n)$	Pointer at grid point (m,n) and level l to initial value e_{l-1} at level $l-1$ from which best path to (m,n) came.
$\tilde{F}_l(m)$	Position along the test pattern, at frame m and level l, from which the best path to the grid point (m, N_v) came.

Manuscript received April 30, 1980; revised September 17, 1980. The work performed by C. S. Myers was done during an M.I.T. cooperative assignment at Bell Laboratories.

C. S. Myers is at the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.

L. R. Rabiner is with Bell Laboratories, Murray Hill, NJ 07974.

$\tilde{F}_l^B(m)$	Value of $\tilde{F}_l(m)$ associated with the reference word v that gave $\tilde{D}_l^B(m)$.
$D(b, e, v)$	DTW distance between reference pattern R_v and the portion of the test pattern between frame b and frame e.
$\hat{D}(b, e)$	Best DTW distance between any reference pattern and the test pattern between frames b and e.
$\hat{V}(b, e)$	Reference pattern giving best DTW distance for the portion of the test pattern between frames b and e.
G_l	The global search region at the lth level in the level building algorithm.
$\overline{N}$	Average length of a reference pattern.
S_l^1	Initial frame of reduced search range at the start of the lth level.
S_l^2	Final frame of reduced search range at the start of the lth level.
$c(m)$	Local minimum of $D_l(m-1, n)$ at $(m-1)$st frame of the test.
ϵ	Range parameter for local minimum search.
$\hat{T}(m)$	Distance threshold on accumulated distance at the mth frame of the test.
T_{MIN}	Minimum value of threshold on distance.
T_{MAX}	Slope of distance threshold curve.
δ_{END}	Variable region at end of test pattern within which warp can end.
δ_{R_1}	Variable region at beginning of reference pattern within which warp can start.
δ_{R_2}	Variable region at end of reference pattern within which warp can end.

I. Introduction

SINCE its introduction into the speech recognition area by Sakoe and Chiba in 1971 [1], the technique of dynamic time warping (DTW) has found widespread use in a variety of speech recognition applications [2]-[13]. This algorithm has proven itself reliable and robust for a wide variety of isolated word recognition systems. Recently, extensions of the algorithm have been investigated for application to the problem of connected word recognition [9]-[13]. The results obtained in connected digit recognition studies have been sufficiently promising to justify intensified study into the performance and implementation of an efficient algorithm to solve this problem. It is the purpose of this paper to present a level building dynamic time warping algorithm that can be applied to connected word recognition problems. The algorithm will be shown to be an efficient implementation of the two-level algorithm proposed by Sakoe [10], and is more general but as efficient as the "sampling" algorithm proposed by Rabiner and Schmidt [11].

The proposed level building algorithm is a special case of the stack decoding algorithm of Bahl and Jelinek [13], which has been proposed for use in a continuous speech recognition system. Although there are several implementational differences between the algorithm of Bahl and Jelinek and the one proposed here, the major differences are in terms of emphasis and presentation. Bahl and Jelinek used a probabilistic finite state machine model for the input and used information theory to formulate a probabilistically optimum decoding of the output. In this paper we use a signal processing formulation, leading to a series of algorithms which progressively lead to the full level building DTW algorithm. Included in our presentation is a full discussion of the implementational aspects of the algorithm including the way in which backtracking is carried out—a major step in the method. In order to keep the presentation rigorous, we have developed a complete, but often cumbersome, notation which allows the reader to link together the mathematic concepts with the implementation. We know of no significantly simpler notation which allows us to accomplish these tasks. In a companion paper a simplified verbal description of the level building algorithm is provided which gives a more intuitive description of how this method works [14].

For readers who are already familiar with the stack model of Bahl and Jelinek, it is worthwhile enumerating some of the general relationships between the notations of the level building (LB) algorithm (as described in this paper), and the stack algorithm. These include the following.

1) The test pattern $T(m)$ of the LB algorithm corresponds to the output sequence Y of the stack algorithm.

2) The frames of the reference pattern, $R(n)$, of the LB algorithm correspond to the set of states, S, of the stack algorithm.

3) The accumulated distance vector $\tilde{D}_l(m), m = 1, 2, \cdots, M$ of the LB algorithm corresponds to the state vector $Q(Y/X)$ at the corresponding confluence node.

With these correspondences in mind, the reader should be able to generalize other aspects of the two algorithms.

Before presenting a formal derivation of the new DTW algorithm, it is worthwhile reviewing the differences between connected word recognition and continuous speech recognition. For the connected word recognition problem, it is assumed that a set of reference patterns are available for each unit of the vocabulary (generally the units are isolated words), and that the connected word pattern can be matched by an abutted sequence of isolated reference patterns. In this case the purpose of the DTW algorithm is to provide the optimum time alignment between the spoken input and the sequence of abutted reference patterns, as well as to determine (in an efficient manner) which of the many possible strings of reference patterns best matches the spoken input. For continuous speech recognition, there is generally no fixed set of reference patterns, and, instead of dynamic time warping for alignment, a segmentation and labeling scheme is used to provide an estimate as to the spoken input utterance. Hence, major differences exist in both concept and implementation for connected word and continuous speech recognition systems.

The outline of this paper is as follows. In Section II we review the connected word recognition problem, and set up a general DTW solution to the problem of comparing a *given* abutted word string to the spoken input string. In Section III we present the level building DTW algorithm and show how it can be used to choose the string that provides the best possible match to the spoken input string. A highly efficient implementation of the algorithm is given in this section. In Section IV we compare the level building DTW algorithm to the two-level method proposed by Sakoe [10], and show that, by appropriately rearranging the order of minimizations, the two-level algorithm becomes the level building DTW algorithm. In Section V we show how modifications can be made in the implementation of the level building DTW algorithm to increase the efficiency and accuracy of the method. Finally, in Section VI we compare the level building DTW algorithm to the methods proposed by Sakoe [10], and Rabiner and Schmidt [11] in terms of computation and storage. The advantages of the new algorithm will be seen in these comparisons.

II. Application of Dynamic Time Warping to Connected Word Strings

Assume we have an unknown test pattern, $T(m)$, $m = 1, 2, \cdots, M$ where, for each value of m, $T(m)$ denotes a vector of features, and M is the length of T in frames. We also assume that T is to be time registered with a sequence of L reference patterns $R_{q(1)}(n)$, $R_{q(2)}(n), \cdots, R_{q(L)}(n)$, where each R_{qk}, $k = 1, 2, \cdots, L$ is one of a set of V reference patterns R_v, $v = 1, 2, \cdots, V$. The length of the vth reference pattern is denoted as N_v. As suggested by Sakoe [10], we define a "super" reference pattern, $R^s_{q(1)q(2)\cdots q(L)}$ (which will be denoted as R^s when there is no ambiguity), as the concatenation of the L reference patterns $R_{q(1)}, R_{q(2)}, \cdots, R_{q(L)}$, i.e.,

$$R^s = R_{q(1)} \oplus R_{q(2)} \oplus R_{q(3)} \cdots \oplus R_{q(L)} \tag{1a}$$

or

$$R^s(n) = \begin{cases} R_{q(1)}(n - \phi(0)), & 1 + \phi(0) \leq n \leq \phi(1) \\ R_{q(2)}(n - \phi(1)), & 1 + \phi(1) \leq n \leq \phi(2) \\ \vdots & \\ R_{q(l)}(n - \phi(l-1)), & 1 + \phi(l-1) \leq n \leq \phi(l) \\ \vdots & \\ R_{q(L)}(n - \phi(L-1)), & 1 + \phi(L-1) \leq n \leq \phi(L) \end{cases} \tag{1b}$$

where the length function $\phi(l)$ is defined as

$$\phi(l) = \sum_{k=1}^{l} N_{q(k)} \tag{2a}$$

$$\phi(0) = 0 \tag{2b}$$

The time registration of the test pattern T, with a "super" reference pattern R^s can now be proposed as a dynamic time warping alignment problem as illustrated in Fig. 1. The optimum time alignment path is denoted by w and we have

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

as the functional mapping between test frame m and super reference frame n. Since R^s can be treated as a single reference pattern (when $q(l)$ is known for all l), time alignment between $T(m)$ and $R^s(n)$ (i.e., determination of w) can be accomplished using a *single application* of a constrained endpoint DTW algorithm.

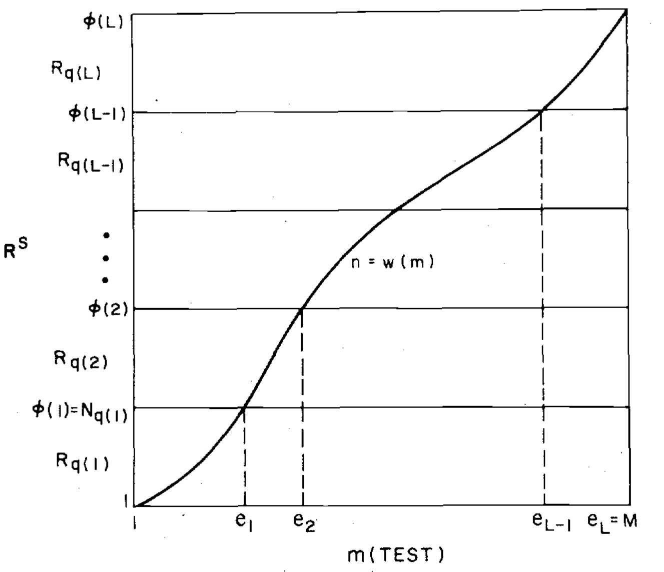

Fig. 1. Illustration of dynamic warping alignment between test pattern T and super reference pattern R^S.

If we denote the local distance between the mth frame of the test pattern, and the nth frame of the super reference as $d(m, n)$, then the constrained endpoint DTW minimizes the global distance D, defined as

$$D = \min_{w(m)} \left[\sum_{m=1}^{M} d(m, w(m)) \right] \tag{4}$$

subject to the endpoint constraints

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

$$w(M) = \phi(L) \tag{5b}$$

and to prescribed local path constraints on $w(m)$. We will assume, for the moment, that $w(m)$ is any nondecreasing function, but later we show how restrictions on the slope of $w(m)$ may be incorporated into the level building algorithm. Although the DTW algorithm for minimizing (4) is well-known [2], we summarize the steps in its implementation because of their importance to the level building DTW algorithm to be presented here. The steps in the method are the following.

Algorithm 1–Constrained Endpoint DTW Algorithm

1) Initialize $D_A(0, 0) = 0, D_A(0, n) = \infty, n \neq 0$.

2) Recursively, for $m = 1, 2, \cdots, M$, and for $n = L(m), \cdots, U(m)$, compute

$$\hat{n} = \underset{\tilde{L}(n) \leq n' \leq \tilde{U}(n)}{\operatorname{argmin}} [D_A(m-1, n')]$$

$$D_A(m, n) = d(m, n) + D_A(m-1, \hat{n}).$$

3) Final solution $D = D_A(M, \phi(L))$

$\underset{x}{\operatorname{argmin}} f(x)$ is the value of x that minimizes $f(x)$, $D_A(m, n)$ is the accumulated distance along the best path from the point ($m = 1$, $n = 1$) to the mth frame of the test pattern and the nth frame of the super reference pattern, $L(m)$ and $U(m)$ are "lower" and "upper" boundary functions on the values of n for a given value of m in order to restrict the range of w to fall within a reasonable set of the (m, n) plane, and where $\tilde{L}(n)$ and $\tilde{U}(n)$ are lower and upper restrictions on the incremental slope of $w(m)$, i.e., $w(m) - w(m-1)$. Specification of $L(m)$, $U(m)$, $\tilde{L}(n)$, and $\tilde{U}(n)$ are unnecessary, at this point, for understanding the implementation of the DTW algorithm, and we only assume that $w(m)$ is nondecreasing, i.e., $\tilde{U}(n) \leq n$.

III. A Level Building Approach to Dynamic Time Warping

The most important feature of Algorithm 1, as implemented above, is not the specific sequence of steps, but the observation that the algorithm can be implemented in levels, i.e., one reference (of the super reference pattern) at a time. To see how this can be done, we first define $D_l(m, n)$ as the local accumulated distance to the mth frame of the test, and the nth frame of the lth reference in the super reference pattern, i.e.,

$$D_l(m, n) = D_A(m, n + \phi(l-1)). \tag{6}$$

We next define $\tilde{D}_l(m)$ as the accumulated distance to the mth frame of the test, and the last frame of the lth reference digit, i.e.,

$$\tilde{D}_l(m) = D_l(m, N_{q(l)}) = D_A(m, \phi(l)) \tag{7a}$$

$$\tilde{D}_0(m) = \begin{cases} 0 & \text{for } m = 0 \\ \infty & \text{for } m \neq 0. \end{cases} \tag{7b}$$

We now can compute the optimum path in levels as follows.

Algorithm 2—Constrained Endpoint DTW Using Level Solutions

```
1) For $l = 1, 2, \cdots, L$, do steps 2)–4).
2) Initialize $D_l(m, 0) = \tilde{D}_{l-1}(m)$, $m = 0, 1, \cdots, M$.
3) Recursively, for $m = 1, 2, \cdots, M$, and for $n = L_l(m), \cdots, U_l(m)$, compute
       $\hat{n} = \underset{\tilde{L}(n) \leq n' \leq \tilde{U}(n)}{\operatorname{argmin}} [D_l(m-1, n')]$
       $D_l(m, n) = D_l(m-1, \hat{n}) + d_l(m, n)$.
4) $\tilde{D}_l(m) = D_l(m, N_{q(l)})$, $m = 1, 2, \cdots, M$.
5) $D = \tilde{D}_L(M)$.
```

$d_l(m, n)$ is the local distance between the mth frame of the test, and the nth frame of the lth reference, i.e.,

$$d_l(m, n) = d(m, n + \phi(l-1)) \tag{8}$$

and $L_l(m)$ and $U_l(m)$ are modified lower and upper boundary functions to account for the lth level, i.e.,

$$L_l(m) = \max(1, L(m) - \phi(l-1)) \tag{9a}$$

$$U_l(m) = \min(N_{q(l)}, U(m) - \phi(l-1)). \tag{9b}$$

A graphical description of Algorithms 1 and 2 is given in Fig. 2. For Algorithm 1 (shown at the top of the figure) the computation is done in vertical strips (i.e., variable n for each m) which generally include frames from two or more references. For Algorithm 2 (shown at the bottom of the figure) the computation is done in a sequence of vertical strips *within* each reference. Following all the vertical strips for the lth reference, the set of accumulated distances along the horizontal line $n = \phi(l)$ is saved (shown by the heavy dots), and used as a set of initial distances for the next level.

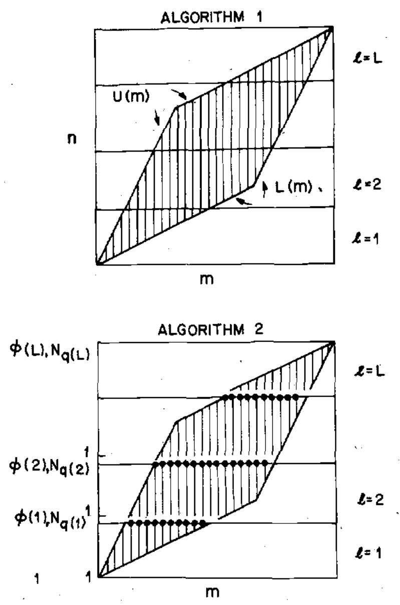

Fig. 2. Graphical description of the computations of Algorithm 1 and 2 showing the detailed manner in which the computation is performed.

A. *Extension of the Level Building Algorithm to Multiple Reference Patterns*

In the preceding section we showed that a given super reference pattern $\boldsymbol{R}^s$ could be time aligned with a test pattern $\boldsymbol{T}$ using a level building DTW algorithm. We now show how this same approach can be applied to a set of variable reference patterns—i.e., when each reference of the super reference pattern is one of a set of V reference patterns.

Before showing how this task is accomplished, several points must be made. First, we define a set of "test" endpoints, e_l, such that,

$$w(e_l) = \phi(l), \tag{10}$$

i.e., e_l is the value of m (along the test pattern) where the optimal path w maps e_l to $\phi(l)$, the end of the lth reference pattern. Fig. 1 illustrates the relation between e_l and $\phi(l)$ for a simple example. If $w(m)$, the warping path, is known, we can determine the values e_l as

$$e_l = w^{-1}(\phi(l)), \quad l = 1, 2, \cdots, L. \tag{11}$$

Since we will be interested in comparing the distance scores of a set of super reference patterns compared to a given test, it is necessary to normalize the values of D. However, from (4), we see that the normalization factor is M, the length of the test, and this is independent of the super reference pattern. Thus, it is not necessary to normalize the values of D for two different super reference patterns in order to compare their distances.

Another point that we must keep in mind is that, in general, the length of the super reference pattern, L, is not known. Generally all that is known is a set of bounds on L of the form

$$L_{\text{MIN}} \leq L \leq L_{\text{MAX}}. \tag{12}$$

Hence, we must worry about variable length super reference patterns as well.

Finally, for a given super reference pattern $R^s_{q(1)q(2)\cdots q(L)}$ we define the distance $D_{q(1)q(2)\cdots q(L)}(m)$ as the distance provided by DTW Algorithm 2 when matching the entire super reference R^s to the portion of the test pattern between frame 1 and frame m. Thus, $D = D_{q(1)q(2)\cdots q(L)}(M)$.

We can now state the connected word recognition problem as the solution to the minimization

$$\min_{L_{\text{MIN}} \leq L \leq L_{\text{MAX}}} \left[\min_{q(1), q(2), \cdots, q(L)} (D_{q(1)q(2)\cdots q(L)}(M)) \right], \tag{13}$$

i.e., minimize the distance D over all possible super reference patterns of all possible lengths. The sequence $q(1), q(2), \cdots, q(L)$ which minimizes (13) is chosen as the estimated test pattern.

In order to solve (13) efficiently, we wish to use a level building approach of the type described in Algorithm 2, i.e., where we decide on the proper lth reference pattern before attempting to find the proper $(l+1)$st reference pattern. To show how this can be done, the following theorem is needed.

Theorem: If there exists a super reference pattern $R^s_{r_1 r_2 \cdots r_l}$, such that,

$$D_{r_1 r_2 \cdots r_l}(e_l) \leq D_{s_1 s_2 \cdots s_l}(e_l),$$

for any $R^s_{s_1 s_2 \cdots s_l}$, then $D_{r_1 r_2 \cdots r_l r_{l+1}}(e_{l+1}) \leq D_{s_1 s_2 \cdots s_l r_{l+1}} \cdot (e_{l+1})$ for any $R^s_{s_1 s_2 \cdots s_l}$ and any $R_{r_{l+1}}$. (The proof of this theorem is given in the Appendix.)

This theorem may be interpreted as saying that if the best sequence of l reference patterns to match the test up to frame e_l is $R_{r_1} \oplus R_{r_2} \oplus \cdots \oplus R_{r_l}$, then this set of l reference patterns will be the first l reference patterns of the best sequence of length $l+1$ where the $(l+1)$st reference pattern corresponds to the portion of the test pattern between frame $(e_l + 1)$ and frame e_{l+1}. (Note that if $R^s_{r_1 \cdots r_l}$ and $R^s_{s_1 \cdots s_l}$ end at different frames along the test then the above theorem need not be true.)

Using the above theorem, we can now find the optimal string of length L as follows.

Algorithm 3

```
1) For $l = 1, 2, \cdots, L_{\text{MAX}}$ do steps 2 and 3.
2) For $v = 1, 2, \cdots, V$, compute
   $D_{r_1 r_2 \cdots r_{l-1} v}(e_l)$ from steps 2)-4) of Algorithm 2.
3) $r_l = \operatorname{argmin}_{1 \leq v \leq V} [D_{r_1 r_2 \cdots r_{l-1} v}(e_l)]$.
4) $\tilde{R}^s_L = R^s_{r_1 r_2 \cdots r_L}$.
```

$\tilde{R}^s_L$ is the best super reference of length L. It is readily seen that, at every level l, this algorithm finds the reference pattern which minimizes the incremental distance at that level.

Algorithm 3, however, has a major flaw—namely, it assumes that the test endpoints $e_1, e_2, \cdots, e_L$ are known *at each* level. This, however, is not the case since the e_l are not determined until the entire path is known. Hence, Algorithm 3 must be modified so that, for all possible sets of $e_1, e_2, \cdots, e_L$, the best sequence of L reference patterns is determined. To do this we define $\tilde{D}^B_l(m)$ as the accumulated distance to the mth frame of the test, and the end of the lth reference pattern of the best partial super reference pattern (to frame m). Then we have

$$\tilde{D}^B_l(m) = \min_{q(1)q(2), \cdots, q(l)} [D_{q(1)q(2), \cdots, q(l)}(m)] \tag{14a}$$

with initial values

$$\tilde{D}^B_0(m) = \begin{cases} 0, & \text{for } m = 0 \\ \infty, & \text{for } m \neq 0. \end{cases} \tag{14b}$$

We can now modify Algorithm 3 to give the following.

Algorithm 4

```
1) Initialize $\tilde{D}^B_0(0) = 0$, $\tilde{D}^B_0(m) = \infty$, $m = 1, 2, \cdots, M$, $\tilde{D}^B_l(0) = \infty$, $l = 1, 2, \cdots, L_{\text{MAX}}$.
2) For $l = 1, 2, \cdots, L_{\text{MAX}}$ do steps 3) through 7).
3) For $v = 1, 2, \cdots, V$ do steps 4) through 6).
4) Initialize $D_l(m, 0) = \tilde{D}^B_{l-1}(m)$, for $m = 0, 1, \cdots, M$.
5) Compute using $R_{q(l)}$ (at the $l$th stage), for $m = 1, 2, \cdots, M$, $n = L_l(m), \cdots, U_l(m)$
   $\hat{n} = \operatorname{argmin}_{\tilde{L}(n) \leq n' \leq \tilde{U}(n)} [D_l(m-1, n')]$
   $D_l(m, n) = d_l(m, n) + D_l(m-1, \hat{n})$.
6) $\tilde{D}^v_l(m) = D_l(m, N_v)$.
7) $\tilde{D}^B_l(m) = \min_{1 \leq v \leq V} [\tilde{D}^v_l(m)]$, $m = 1, 2, \cdots, M$.
8) $W_l(m) = \operatorname{argmin}_{1 \leq v \leq V} [\tilde{D}^v_l(m)]$, $m = 1, 2, \cdots, M$.
9) $L = \operatorname{argmin}_{L_{\text{MIN}} \leq l \leq L_{\text{MAX}}} [\tilde{D}^B_l(M)]$
   $D = \tilde{D}^B_L(M)$.
10) $\tilde{R}^s = R_{W_1(e_1)} \oplus R_{W_2(e_2)} \oplus \cdots \oplus R_{W_L(e_L)}$.
```

The array $W_l(m)$ records the index v of the reference pattern R_v which gives the best path to the end of the lth level at the mth frame of the test. The array $\tilde{D}^v_l(m)$ is the accumulated distance to the end of the lth level using reference pattern R_v at the frame m of the test.

Algorithm 4 may be looked at as building up all possible paths, one reference pattern at a time. Step 2) initiates building paths by levels, l. Step 3) loops through all reference patterns at level l. Steps 4)-6) perform the dynamic programming (DP) recursion for a fixed reference pattern $q(l)$ at level l. (This is similar to the computations of the level building of Algorithm 2.) Steps 7) and 8) finds the best path for all possible references at the end of the lth level, and records the reference pattern R_v associated with the best path. Step 9) chooses the best length for the best super reference pattern, and step 10) recovers the best super reference pattern.

Fig. 3 illustrates a typical example of the use of Algorithm 4. For simplicity we assume that $L = 4$ is known, that $w(m)$ is restricted to be in the parallelogram shown in Fig. 3, and that $V = 2$, i.e., only two reference patterns exist. We denote the two reference patterns as A and B. This figure shows, at each level, the best reference pattern associated with the mth frame, $R_{W_l}(m)$, and the best path from the previous level. For example, at the first level, the reference pattern can end at any

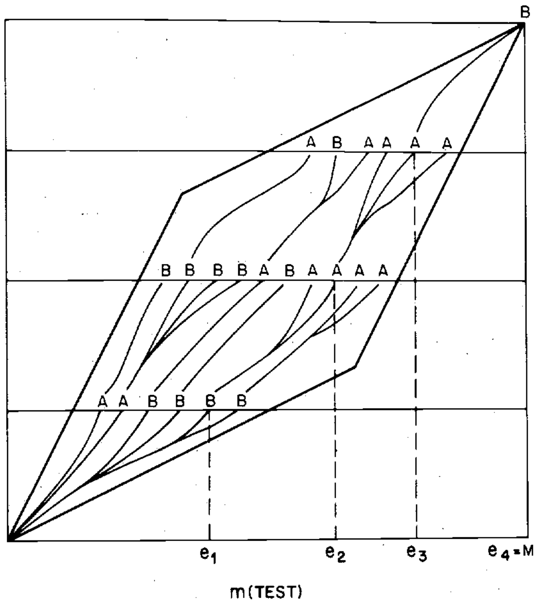

Fig. 3. Typical example of building up warping paths for a two reference case.

one of six different frames along the test. If it ends at either of the first two frames then the best reference is A, and if it ends at any of the other four locations the best reference is B. At each level, for all possible ending frames, the best reference is obtained, until at level four a unique best reference is obtained. To obtain the best super reference pattern, we trace back from the end of the test, giving the pattern $\tilde{\boldsymbol{R}}^s = \boldsymbol{R}_B \oplus \boldsymbol{R}_A \oplus \boldsymbol{R}_A \oplus \boldsymbol{R}_B$, with endpoints e_1, e_2, e_3, and $e_4 = M$ along the test pattern.

B. Implementation of Backtracking

The last step in Algorithm 4 recovers the optimum super reference string $\tilde{\boldsymbol{R}}^s$ by concatenating patterns from a set of test pattern endpoints, e_l, $l = 1, 2, \cdots, L$. In order to obtain the set of endpoints, some form of backtracking must be added to Algorithm 4. The backtracking information must be recorded during the computation of $D_l(m, n)$ because the best path to the point (m, n), at the lth level, is unique–hence, there is a unique ending point e_{l-1} associated with each grid point (m, n). Fig. 4 illustrates this key point. It is assumed that the optimum path to the point (m,n) at the lth level came from the point $(e_{l-1}, \phi(l-1))$ at the end of the $(l-1)$st level. Hence, we define a "from frame" array $F_l(m, n)$ as the value of e_{l-1} from which the best path to (m, n), at the lth level, using reference $q(l)$, came. Then we can backtrack a fixed super reference pattern (Algorithm 2) by the following steps.

Backtracking Algorithm

```
1) For $l = 1, 2, \cdots, L$, do steps 2)-4).
2) For $m = 0, 1, \cdots, M$, initialize $F_l(m, 0) = m$.
3) For $m = 1, 2, \cdots, M$, $n = L_l(m), \cdots, U_l(m)$, compute $\hat{n}$ and $D_l(m, n)$ as in step 3), Algorithm 2, and compute $F_l(m, n) = F_l(m-1, \hat{n})$.
4) $\tilde{F}_l(m) = F_l(m, N_v)$, $m = 1, 2, \cdots, M$.
5) $e_L = M$.
6) $e_l = \tilde{F}_{l+1}(e_{l+1})$, $l = L-1, L-2, \cdots, 1$.
```

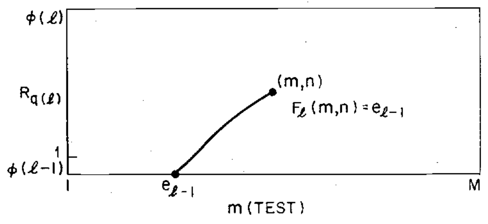

Fig. 4. Backtracking pointer from grid point (m, n) to initial value e_{l-1} at the lth level.

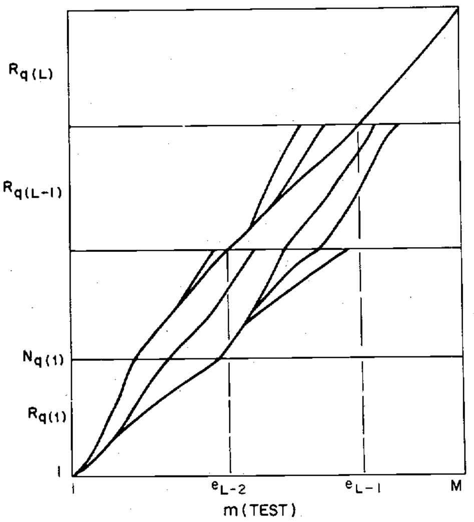

Fig. 5. Example of time alignment with several local paths.

$\tilde{F}_l(m)$ keeps track of the position along the test at the end of the $(l-1)$st level from which the best path to (m, N_v) came. $F_l(m,n)$ is computed as $F_l(m-1, \hat{n})$ because the best path to (m, n) came from the same place as the best path to each intermediate point of the path, namely $(m-1, \hat{n})$. Fig. 5 shows an example of a time alignment along with several local paths. It can be seen that $\tilde{F}_L(e_L)$ is e_{L-1}, and that, using the $\tilde{F}_l(m)$ array, it is possible to trace back the endpoints at each level.

With the above discussion, it is fairly straightforward to apply a backtracking procedure to Algorithm 4 (i.e., when the super reference patterns are variable). We define $\tilde{F}_l^B(m)$ as the value of $\tilde{F}_l(m)$ associated with the best reference word $W_l(m)$ (smallest distance) at the lth level, and with position m of the test. We also define $\tilde{F}_l^v(m)$ as the value of $\tilde{F}_l(m)$ for reference $\boldsymbol{R}_v$. Thus, the total level building DTW algorithm becomes Algorithm 5.

Algorithm 5–Level Building DTW Algorithm

```
1) Initialize $\tilde{D}_0^B(m) = 0$, for $m = 0$, $\tilde{D}_0^B(m) = \infty$, $m = 1, 2, \cdots, M$, $\tilde{D}_l^B(0) = \infty$, $l = 1, 2, \cdots, L_{\text{MAX}}$.
2) For $l = 1, 2, \cdots, L_{\text{MAX}}$ do steps 3) through 7).
3) For $v = 1, 2, \cdots, V$, do steps 4) through 6).
4) For $m = 0, 1, \cdots, M$, initialize $D_l(m, 0) = \tilde{D}_{l-1}^B(m)$, $F_l(m, 0) = m$.
5) For $m = 1, 2, \cdots, M$, and $n = L_l(m), \cdots, U_l(m)$ compute using $\boldsymbol{R} = \boldsymbol{R}_v$.
```

$$\hat{n} = \underset{\tilde{L}(n) \leq n' \leq \tilde{U}(n)}{\text{argmin}} [D_l(m-1, n')]$$

$$D_l(m, n) = d_l(m, n) + D_l(m-1, \hat{n})$$

$F_l(m, n) = F_l(m - 1, \hat{n})$.

6) For $m = 1, 2, \cdots, M$, compute

$\tilde{D}_l^v(m) = D_l(m, N_v)$

$\tilde{F}_l^v(m) = F_l(m, N_v)$.

7) For $m = 1, 2, \cdots, M$, compute

$\tilde{D}_l^B(m) = \min\limits_{1 \leq v \leq V} [\tilde{D}_l^v(m)]$

$W_l(m) = \operatorname*{argmin}\limits_{1 \leq v \leq V} [\tilde{D}_l^v(m)]$

$\tilde{F}_l^B(m) = \tilde{F}_l^{W_l(m)}(m)$.

8) $L = \operatorname*{argmin}\limits_{L_{\text{MIN}} \leq l \leq L_{\text{MAX}}} [\tilde{D}_l^B(M)]$

$D = \tilde{D}_L^B(M)$.

9) $e_L = M$.

10) $e_l = \tilde{F}_{l+1}^B(e_{l+1}), l = L - 1, L - 2, \cdots, 1$.

11) $\tilde{R}^s = R_{W_1(e_1)} \oplus R_{W_2(e_2)} \oplus \cdots \oplus R_{W_L(e_L)}$.

Algorithm 5 represents a complete, level building approach to optimally matching a concatenated set of reference patterns (with variable references) to a given test pattern. The output of the algorithm is the optimal number of reference patterns, L, the minimum string distance D, the references in the string $R_{W_1}(e_1), R_{W_2}(e_2), \cdots, R_{W_L}(e_L)$, and the set of test pattern endpoints, e_l, that match the last frame of reference patterns in the string.

IV. Relationship of Level Building Algorithm 5 to Two-Level DTW Algorithm of Sakoe

In a recent paper, [10] Sakoe proposed an alternative DTW algorithm, called the two-level DP matching algorithm, for obtaining the "best" match to a given test string of words. The algorithm may be summarized as follows.

Algorithm S1 (Sakoe)

I. Word Level Matching

1) Compute word distances $D(b, e, v)$ for $1 \leq b \leq e \leq M$ and for $1 \leq v \leq V$ where $D(b, e, v)$ is the DTW distance between reference pattern R_v and the test pattern, subject to the endpoint constraints $w(b) = 1$, $w(e) = N_v$, i.e., reference R_v matches the portion of the test pattern from frame b to frame e.

2) Compute best beginning-ending distances, $\hat{D}(b, e)$ as

$$\hat{D}(b, e) = \min_{1 \leq v \leq V} [D(b, e, v)] \tag{15a}$$

$$\hat{V}(b, e) = \operatorname*{argmin}_{1 \leq v \leq V} [D(b, e, v)] \tag{15b}$$

where $\hat{V}(b, e)$ keeps track of the best reference pattern that matches the test between frames b and e.

II. Phrase Level Matching

1) Initialize $\tilde{D}_0^B(0) = 0$. (16)

2) Compute, for $e = 1, 2, \cdots, M$, and for $l = 1, 2, \cdots, L_{\text{MAX}}$, the quantities

$$\tilde{D}_l^B(e) = \min_{1 \leq b \leq e} [\hat{D}(b, e) + \tilde{D}_{l-1}^B(b - 1)] \tag{17a}$$

$$\tilde{F}_l^B(e) = \operatorname*{argmin}_{1 \leq b \leq e} [\hat{D}(b, e) + \tilde{D}_{l-1}^B(b - 1)] - 1 \tag{17b}$$

where $\tilde{F}_l^B(e)$ is the ending point, along the test, of the $(l - 1)$st reference pattern in the sequence which generates the minimum distance $\tilde{D}_l^B(e)$.

3) Find best length of sequence, L, as

$$L = \operatorname*{argmin}_{L_{\text{MIN}} \leq l \leq L_{\text{MAX}}} [\tilde{D}_l^B(M)]. \tag{18}$$

4) Set string endpoints as

$$e_L = M \tag{19a}$$

$$e_0 = 0. \tag{19b}$$

5) Backtrack for $l = L - 1, L - 2, \cdots, 1$, to give

$$e_l = \tilde{F}_{l+1}^B(e_{l+1}). \tag{20}$$

6) Recover best reference pattern indices $v_1, v_2, \cdots, v_L$ as

$$v_l = \hat{V}(e_{l-1} + 1, e_l), 1 = 1, 2, \cdots, L. \tag{21}$$

7) Form best string as

$$\tilde{R}_L^s = R_{v_1} \oplus R_{v_2} \oplus \cdots \oplus R_{v_L}. \tag{22}$$

We now show how Algorithm 5, the level building algorithm, can be expressed as a single, but important modification of Sakoe's algorithm. If we substitute (15a) into (17a) we get

$$\tilde{D}_l^B(e) = \min_{1 \leq b \leq e} [\min_{1 \leq v \leq V} (D(b, e, v)) + \tilde{D}_{l-1}^B(b - 1)]. \tag{23}$$

If we reverse the order of the minimizations we get

$$\tilde{D}_l^B(e) = \min_{1 \leq v \leq V} [\min_{1 \leq b \leq e} (D(b, e, v) + \tilde{D}_{l-1}^B(b - 1))]. \tag{24}$$

With the reversed order of minimizations we can rewrite Algorithm S1 to the modified form that follows.

Algorithm S2

I. Word Level Matching

1) For $1 \leq b \leq e \leq M$, and for $1 \leq v \leq V$, compute word distances $D(b, e, v)$.

II. Phrase Level Matching

1) Initialize $\tilde{D}_0^B(0) = 0$.

2) For $l = 1, 2, \cdots, L_{\text{MAX}}$, do steps 3) and 4).

3) For $e = 2, 3, \cdots, M$, and for $v = 1, 2, \cdots, V$, compute first minimization results

$$\tilde{D}_l^v(e) = \min_{1 \leq b \leq e} [D(b, e, v) + \tilde{D}_{l-1}^B(b - 1)] \tag{25a}$$

$$\tilde{F}_l^v(e) = \operatorname*{argmin}_{1 \leq b \leq e} [D(b, e, v) + \tilde{D}_{l-1}^B(b - 1)] - 1 \tag{25b}$$

where $\tilde{D}_l^v(e)$ is the best distance to frame e of the test when the lth reference pattern is R_v, and $\tilde{F}_l^v(e)$ is the frame along the test at which the $(l - 1)$st reference pattern ended.

4) Compute second minimization

$$\tilde{D}_l^B(e) = \min_{1 \leq v \leq V} [\tilde{D}_l^v(e)] \tag{26a}$$

$$\tilde{W}_l(e) = \operatorname*{argmin}_{1 \leq v \leq V} [\tilde{D}_l^v(e)] \tag{26b}$$

$$\tilde{F}_l^B(e) = \tilde{F}_l^{W_l(e)}(e) \tag{26c}$$

where $W_l(e)$ is the reference pattern at level l and test frame e which minimizes $\tilde{D}_l^B(e)$.

5) Find best length L as

$$L = \underset{L_{\mathrm{MIN}} \leq l \leq L_{\mathrm{MAX}}}{\mathrm{argmin}} [\tilde{D}_l^B(M)]. \tag{27}$$

6) Set endpoints

$$e_L = M. \tag{28}$$

7) Backtrack for $l = L - 1, L - 2, \cdots, 1$, giving

$$e_l = \tilde{F}_{l+1}^B(e_{l+1}). \tag{29}$$

8) Recover best reference patterns as

$$v_l = W_l(e_l), l = 1, 2, \cdots, L. \tag{30}$$

9) Form best reference string as

$$R_l^s = R_{v_1} \oplus R_{v_2} \oplus \cdots \oplus R_{v_L}. \tag{31}$$

Algorithm S2 is important because the computation of $\tilde{D}_l^v(e)$ can be made for *all* e (i.e., $1 \leq e \leq M$), and for a fixed v (reference pattern) in a *single* time warp, given $\tilde{D}_{l-1}^B(m)$, $m = 0, 1, \cdots, M$, *without* having to compute $D(b, e, v)$, separately. This can be accomplished as follows.

Algorithm S3–[Replacement for steps I-1), and II-2) and 3)]

1) Initialize, for $m = 0, 1, \cdots, M$,

$$D_l(m, 0) = \tilde{D}_{l-1}^B(m) \tag{32a}$$

$$F(m, 0) = m. \tag{32b}$$

2) For $m = 1, 2, \cdots, M$, and $n = L(m), \cdots, U(m)$, (where N_v is the length of the vth template) recursively compute

$$\hat{n} = \underset{\tilde{L}(n) \leq n' \leq \tilde{U}(n)}{\mathrm{argmin}} [D_l(m-1, n')] \tag{33}$$

$$D_l(m, n) = D_l(m-1, \hat{n}) + d_l(m, n) \tag{34a}$$

$$F_l(m, n) = F_l(m-1, \hat{n}). \tag{34b}$$

3) For $e = 2, 3, \cdots, M$, get final output

$$\tilde{D}_l^v(e) = D_l(e, N_v) \tag{35a}$$

$$\tilde{F}_l^v(e) = F_l(e, N_v). \tag{35b}$$

It can be seen that the computation of Algorithm S3 along with steps 2)-9) of Algorithm S2 are essentially identical to the computation of Algorithm 5 of the previous section, thus completing the proof that the two-level DP matching algorithm of Sakoe is essentially identical to the level-building algorithm. The reason we have described the level-building algorithm in such detail is because it is an *order of magnitude more efficient* than Sakoe's two-level DP matching algorithm. We will examine issues of computation and storage of the two algorithms in detail in Section VI. First, we devote more attention to implementational aspects of the level building algorithm.

V. Implementational Aspects and Modifications of the Level Building DTW Algorithm

We have shown that a level building DTW approach can be used to match a concatenated series of reference patterns to a connected test string. The basic principles used in formulating the algorithm were backtracking path recovery, and local optimality of the path. By local optimality we mean that the best path from the grid point (1, 1) to the grid point (m, n) that goes through the grid point (m', n') includes, as a subsection, the best path from (1, 1) to (m', n'). Hence, by proceeding from left to right, the algorithm finds the best partial string, at each level, and for each value of m, and recovers the correct path by backtracking from the end of the string.

There are a number of computational aspects of algorithm 5 that should be discussed in order to better understand how this algorithm is used in practice. In this section, we discuss these questions.

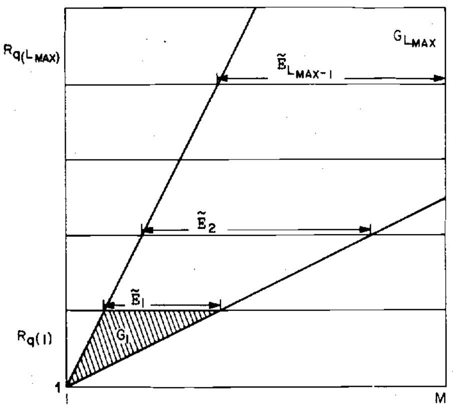

Fig. 6. Illustration of the use of lower and upper boundary constraints on the global region of paths in the level building DTW algorithm.

A. Range Restrictions on n

If we examine the computation of Algorithm 5 for obtaining the accumulated distance at grid point (m, n) at level l, $D_l(m, n)$, we see that the global range on n is restricted by a set of lower and upper boundary constraints, $L(m)$ and $U(m)$. Typically, these boundary constraints restrict the area of the (m, n) plane in which the best path can lie. For example a standard set of global constraints is

$$L(m) = \frac{(m+1)}{2} \tag{36a}$$

$$U(m) = 2(m-1) + 1. \tag{36b}$$

The global set of constraints of (36) is independent of n and merely restricts the path w to lie in a subset of the (m, n) plane, defined by lines of slope $\frac{1}{2}$ and 2 originating at the point (1, 1). Fig. 6 illustrates how such path constraints affect the computation of the level building algorithm. We see that at the first level, the DTW recursion needs to be computed only for the shaded region G_1, and not for the entire (m, n) plane. If we define $\tilde{E}_1$ as the ending region (along the m axis) at the end of the first level, then it should be clear that the width of $\tilde{E}_1$ varies as the length of the reference patterns vary. Hence, the width of the ending region, $\tilde{E}_1$, is a superposition of the widths of the ending region over the set of V reference patterns.

Generally, a set of local path constraints is superimposed on

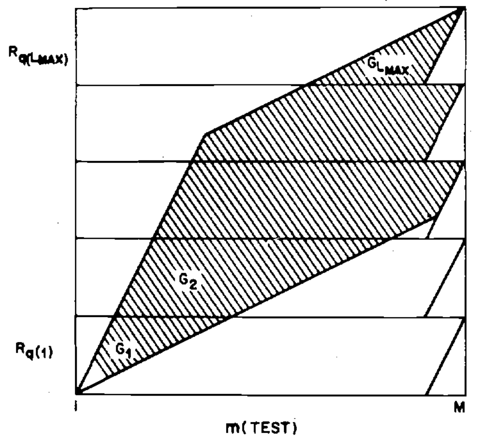

Fig. 7. The global search regions for the level building algorithm with L_{MAX} possible references in the string.

the warping function to ensure that the warping path $w(m)$ does not change drastically as m increases. For example, if we require that

$$0 \leqslant w(m) - w(m-1) \leqslant 2 \tag{37a}$$

$$1 \leqslant w(m) - w(m-2), \tag{37b}$$

i.e., the Itakura constraints, then the upper and lower local constraints become

$$\tilde{L}(n) = \max(0, n-2) \tag{38a}$$

$$\tilde{U}(n) = \begin{cases} n & \text{if } w(m-1) \neq w(m-2) \\ n-1 & \text{if } w(m-1) = w(m-2) \end{cases} \tag{38b}$$

If we examine the size of the global regions, G_l, at the lth level, as shown in Fig. 6, we see that G_l grows continuously with l. It should be clear that as l approaches L, the size of the actual string, the global region G_l should become small since the path must end at the point $(M, \phi(L_{\mathrm{MAX}}))$. As such further restrictions on $L(m)$ and $U(m)$ may be imposed by the boundary conditions at the end of the test. Such restrictions are illustrated in Fig. 7 which shows a line of slope $\frac{1}{2}$ (from the end of the last reference) constraining the upper region of the (m, n) plane. However, unlike Fig. 3 there is no line of slope 2) from the point $(M, \phi(L_{\mathrm{MAX}}))$ since it is not known, *a priori*, whether $L = L_{\mathrm{MAX}}$. Thus, instead of the global line of slope 2) coming from the end of the test, there is a constraint line of slope 2) from each of the possible sets of ending points $(M, \phi(l)), l = L_{\mathrm{MIN}}, \cdots, L_{\mathrm{MAX}}$. These path restrictions, however, are applied only at the given level, since the correct length of the string, L, is not known during the computation of $D_l(m, n)$.

Based on the above discussion, a reasonable set of lower and upper boundary constraints would be of the form

$$L(m) = \max\left[\frac{m+1}{2}, 2(m-M) + \phi(l)\right] \tag{39a}$$

$$U(m) = \min\left[2(m-1)+1, \tfrac{1}{2}(m-M) + \phi(L_{\mathrm{MAX}})\right] \tag{39b}$$

where $\phi(L_{\mathrm{MAX}})$ is the maximum length of the concatenated

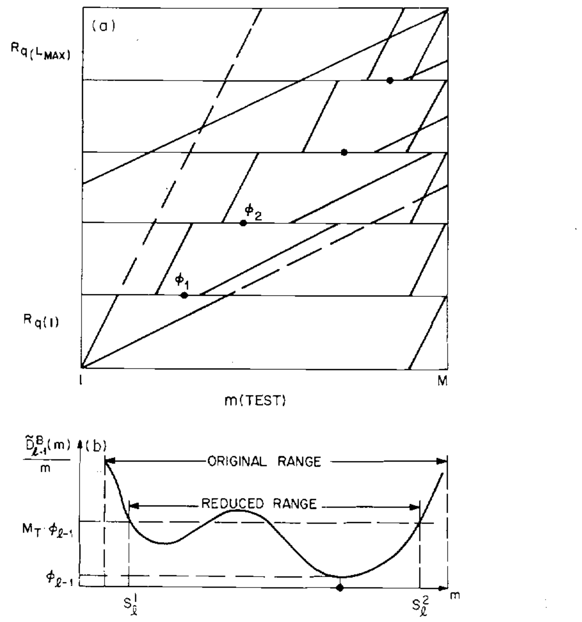

Fig. 8. Illustration of computational technique to reduce search range in the n dimension.

super reference pattern. This region is illustrated as the shaded region in Fig. 7.

It is possible to estimate the area, in the (m, n) plane, covered by the level building dynamic time warping algorithm. For a string of length $L = L_{\mathrm{MAX}}$, and an unconstrained warping function, the average total area is

$$A_T = L_{\mathrm{MAX}} \cdot \bar{N} \cdot M \tag{40}$$

where $\bar{N}$ is the average length of a reference pattern. When the range is restricted to lie in the shaded region of Fig. 7, the average reduced total area is approximately

$$A_R = L_{\mathrm{MAX}} \cdot \bar{N} \cdot M/3 \tag{41}$$

assuming $\bar{N} \approx M/L_{\mathrm{MAX}}$, i.e., the test length is about L_{MAX} times the average reference length. Hence, about a three to one reduction in computation is achieved by using appropriately defined lower and upper boundary constraints.

B. Range Reduction Techniques

Several computational reductions may be incorporated into the level building algorithm. One simple one is to reduce the range at any given level, as illustrated in Fig. 8. Part (a) of this figure shows the original search region (bounded by the dashed lines), and the reduced search region (bounded by the solid lines). The search range is reduced so that regions of the (m, n) plane, where the paths have already accumulated a large distance, are not considered as potential starting points for the next level. The method used to reduce the search range is illustrated in part (b) of Fig. 8. At the beginning of the lth level, the *average* distance to any ending point m at the end of the $(l-1)$st level is computed, as well as, the minimum, ϕ_{l-1},

$$\phi_{l-1} = \min_{1 \leqslant m \leqslant M}\left[\frac{\tilde{D}^B_{l-1}(m)}{m}\right] \tag{42}$$

Based on the value of ϕ_{l-1}, two values S_l^1 and S_l^2 are found via the rule

$$S_l^1 = \text{largest } m \text{ such that } \frac{\tilde{D}_{l-1}^B(m)}{m} > \phi_{l-1} \cdot M_T \text{ for } all\ m < S_l^1 \quad (43a)$$

$$S_l^2 = \text{smallest } m \text{ such that } \frac{\tilde{D}_{l-1}^B(m)}{m} > \phi_{l-1} \cdot M_T \text{ for } all\ m > S_l^2 \quad (43b)$$

where M_T is a parameter controlling the size of the starting range. As M_T approaches one, the values S_l^1 and S_l^2 approach each other, and the width of the reduced range approaches a single point. (This is equivalent to making an unequivocal decision about the correct reference at the end of the $(l-1)$th level; in this manner each level is essentially an independent, constrained endpoint, DTW match.) As M_T approaches ∞, the reduced range is the same as the original range.

Fig. 8(b) illustrates the importance of using a reasonable value of M_T. In this example, the curve of $\tilde{D}_{l-1}^B(m)/m$ has *two* distinct minima. This generally represents two possible ending regions for the $(l-1)$st reference, and any range reduction which eliminates either one of them could potentially lead to errors in obtaining the best string match. This example also illustrates the importance of searching for S_l^1 and S_l^2 from the extremes of the range, since it can be seen that the curve exceeds the threshold between S_l^1 and S_l^2. In this case we do not want to reduce the range to eliminate this region.

It is straightforward to incorporate the range reduction technique into Algorithm 5 by replacing step 4) with the following. For $m = 0, 1, \cdots, M$, initialize

$$D_l(m, 0) = \begin{cases} \tilde{D}_{l-1}^B(m) & \text{if } S_l^1 \leqslant m \leqslant S_l^2 \\ \infty & \text{otherwise} \end{cases}$$

$$F_l(m, 0) = m \qquad \text{if } S_l^1 \leqslant m \leqslant S_l^2$$

where

$$S_l^1 = S_l^2 = 1.$$

In addition, the modified upper and lower boundary functions $U_l(m)$ and $L_l(m)$ are now given by

$$L_l(m) = \max\left[1, \frac{(m - S_l^2)}{2}, 2(m - M) + N_v\right] \quad (44a)$$

$$U_l(m) = \min\left[N_v, 2(m - S_l^1), \frac{(m - M)}{2} + (L_{\text{MAX}} - l) N_{\text{MAX}} + N_v\right] \quad (44b)$$

where N_{MAX} is the length of the largest reference pattern. The last term in (44b) reflects a line of slope $\frac{1}{2}$ coming from the point $m = M$ as in Fig. 7.

Another method of range reduction is illustrated in Fig. 9. Here we show the range reduced by not only shrinking the starting region (along the m-axis), but also by following the local minimum as is done in a local minimum DTW algorithm [7]. In this case the local accumulated distance function $D_l(m, n)$ is computed for $c(m) - \epsilon \leqslant n \leqslant c(m) + \epsilon$, $1 \leqslant m \leqslant M$, where

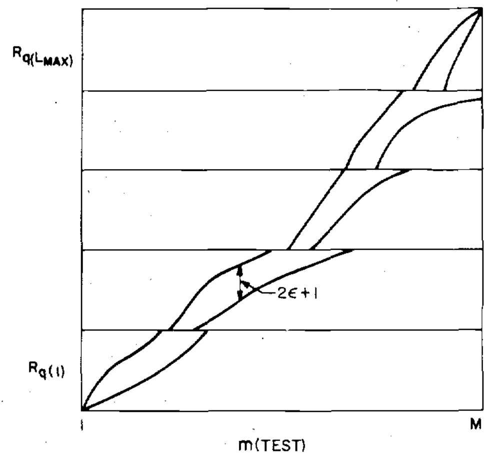

Fig. 9. Illustration of range reduction in the n dimension using a local minimum search.

$$c(m) = \underset{c(m-1)-\epsilon \leqslant m \leqslant c(m-1)+\epsilon}{\text{argmin}} [D_l(m-1, n)] \quad (45a)$$

$$c(1) = 1. \quad (45b)$$

It should be noted that the local minimum algorithm must be modified so that the lower boundary always includes a valid starting point if there is one, i.e., the lower boundary $c(m) - \epsilon$, must be set to 1 if $S_l^1 \leqslant m \leqslant S_l^2$. This constraint is needed to prevent the loss of possible path starting points. Similar local minimum algorithms can be used where ϵ is a variable which is adapted to the data—i.e., ϵ is large when the distance along a vertical strip is close to the local minimum over a broad range, and ϵ is small when the distance along a vertical strip increases rapidly away from the local minimum.

A further reduction in computation may be gained by the use of accumulated distance thresholds [8]. A distance threshold is used to abandon further comparisons for a given reference pattern R_v, at a given level l, when the incremental accumulated distance at the level exceeds a given threshold. We define the average incremental distance to (m, n) at the lth level as

$$\Delta D_l(m, n) = \frac{D_l(m, n) - \tilde{D}_{l-1}^B(F_l(m, n))}{m - F_l(m, n)} \quad (46)$$

whenever $\Delta D_l(m, n)$ exceeds a threshold of the form

$$\hat{T}(m) = T_{\text{MIN}} + T_{\text{MAX}} \cdot (m - F_l(m, n)). \quad (47)$$

The search for the current reference is abandoned. It should be noted that the threshold test cannot be applied until all valid starting points are tried, i.e., until $m > S_l^2$.

C. Test Endpoint Constraint Relaxation

One simple modification to the level building algorithm is to relax the ending condition that $\phi(L) = w(M)$. Rather than forcing the path to end at frame M of the test pattern, more flexibility in the algorithm is achieved if we allow the path to end within the region from $M - \delta_{\text{END}}$ to M. (Clearly if $\delta_{\text{END}} = 0$ we are back to the constrained endpoint case.) The use of a variable ending region is important because it reduces the burden of the endpoint detector (in the recognizer) to find precisely the ending frame of the spoken string of words. The endpoint modification can be implemented by searching for the best string (lowest *normalized* distance) within the modified ending region. Thus, in Algorithm 5 (Section III-B) we replace steps 8) and 9) with the following:

8) $(L, E) = \underset{L_{\text{MIN}} \leqslant l \leqslant L_{\text{MAX}}}{\operatorname{argmin}} \; \underset{M - \delta_{\text{END}} \leqslant e \leqslant M}{\operatorname{argmin}} \; [\tilde{D}_l^B(e)/e]$

9) $e_L = E$

where E is used to record the ending point.

D. Modified (Partial) Reference Patterns

Another modification to the level building algorithm is suggested by the work of Rabiner and Schmidt [11]. They proposed that the time warping not use the entire reference pattern, but instead be allowed to eliminate a portion at the end of the reference pattern. This type of modification accounts for some of the gross features of coarticulation between consecutive reference patterns. Another interpretation is that the reference patterns, obtained generally from isolated occurrences of spoken words, are generally longer than the word spoken in a string, and this type of modification accounts for some of the word shortening in a connected string. To implement this modification, we require that, at any level, the warping function end somewhere between frames $N_v - \delta_{R_2}$ and N_v of the reference pattern, rather than being forced to end at frame N_v, as is normally the case. If the parameter δ_{R_2} is set to 0, then this modification is transparent in the algorithm. In general, one would expect that the maximum number of frames which can be eliminated from the end of the reference pattern, δ_{R_2} should be a function of the reference pattern itself. However, for simplicity, we assume δ_{R_2} is a constant for all reference patterns. To implement a modified reference pattern, step 6) of Algorithm 5 is modified to the following.

6) For $m = 1, 2, \cdots, M$ compute

$$\tilde{n} = \underset{N_v - \delta_{R_2} \leqslant n \leqslant N_v}{\operatorname{argmin}} [D_l(m, n)]$$

$$\tilde{D}_l^v(m) = D_l(m, \tilde{n})$$

$$\tilde{F}_l^v(m) = F_l(m, \tilde{n})$$

where $\tilde{n}$ is used to indicate the best ending point in the reference pattern.

A similar modification can be used in the level building algorithm at the beginning of reference patterns. If we assume that the warping function can begin anywhere from frame 1 to $1 + \delta_{R_1}$ of the reference pattern (where $\delta_{R_1} = 0$ says that we use the whole beginning region) then we need to modify step 5) of Algorithm 5 to give the following.

5) For $m = 1, 2, \cdots, M$ and for $n = L_l(m), \cdots, U_l(m)$, compute using $R = R_v$

$$\hat{n} = \begin{cases} \underset{n' = 0, \text{ or } \tilde{L}(n) \leqslant n' \leqslant \tilde{U}(n)}{\operatorname{argmin}} [D_l(m-1, n')] & \text{for } 1 \leqslant n \leqslant 1 + \delta_{R_1} \\ \underset{\tilde{L}(n) \leqslant n' \leqslant \tilde{U}(n)}{\operatorname{argmin}} [D_l(m-1, n')] & \text{for } n > 1 + \delta_{R_1} \end{cases}$$

$$D_l(m, n) = d_l(m, n) + D_l(m-1, \hat{n})$$

$$F_l(m, n) = F_l(m-1, \hat{n}).$$

The modified local constraints say that if n is within the expanded starting region of the reference pattern (within the first $\delta_{R_1} + 1$ frames), then the previous accumulated distance can come from the initial conditions at the end of the previous level.

E. Multiple Candidate Strings

For many applications of connected word recognition, it is required that more than one super reference pattern be given as a possible recognition candidate. The level building algorithm, as described here, is not directly applicable to obtain the best Q sets of super reference patterns because only the information on best candidates is retained. One reasonable way to generate a list of plausible candidates is to retain the data (distances, backtracking pointers, etc.) on both the best and second best choice at every level and every ending point. To generate alternate candidates, the normal backtracking procedure is followed, but second best references are inserted in place of first best references at the different levels. It should be noted that this technique is not guaranteed to generate the best Q candidates; however, it does give a reasonable list of candidates.

F. Use of Syntactical Constraints

The level building algorithm, as described in this paper, can only be applied to an unrestricted set of super references—i.e., without syntactical constraints among the references in the string. It is easy to modify the level building algorithm, however, if we observe that, just as the best way to any level does not depend on any future level, the best way to any state in a finite state machine (FSM) (with no loops) depends only on past states. We shall assume that the syntax may be represented by a FSM M (S, V, Δ, s_0, Z) where S is a set of states $\{s_0, s_1, \cdots, s_{\beta-1}\}$ $(\beta = |S|)$, V is a vocabulary (labeled $1, 2, \cdots, \gamma$), $(\gamma = |V|)$, Δ is the set of allowable transitions, i.e., $\Delta: S \times V \rightarrow S$ $s_0 \in S$ is the initial state, and $t \subset Z$ is the set of final states. Furthermore, we assume that M has no loops and that the states $s_0, s_1, \cdots, s_{\beta-1}$ are ordered so that if $i \geqslant j$ then there is no path from s_i to s_j. Thus, the best path to state $s_{i'}$ and frame m of the test depends only of states s_i, for $0 \leqslant i \leqslant i'$. In this case, the algorithm is similar to the level building algorithm with the following exceptions.

TABLE I-A
COMPUTATIONAL COMPARISONS OF CONNECTED WORD DTW ALGORITHM

	Level Building	Two-Level DP Warp	Sampling	Reduced Level Building
Number of Basic Time Warps	$L_{MAX} \cdot V$	$M \cdot V$	$L \cdot V \cdot \bar{\gamma}$	$L \cdot V$
Size of Time Warps	$\bar{N} \cdot M/3$	$\bar{N} \cdot (2R+1)$	$\bar{N} \cdot (2\bar{\epsilon}+1)$	$\bar{N} \cdot (2\epsilon+1)$
Total Computation For Distances	$L_{MAX} \cdot V \cdot \bar{N} \cdot M/3$	$M \cdot V \cdot \bar{N} \cdot (2R+1)$	$L \cdot V \cdot \bar{\gamma} \cdot \bar{N}\,(2\bar{\epsilon}+1)$	$L \cdot V \cdot \bar{N}\,(2\epsilon+1)$
Storage	$3 \cdot M \cdot L_{MAX}$	$2 \cdot M \cdot (2R+1)$	0	$3 \cdot M \cdot L_{MAX}$

1) The results are built up by states $s_0, s_1, \cdots, s_{\beta-1}$, rather than by levels.

2) In building up to a particular state, instead of using all references of the vocabulary, we use all *transitions* to that state.

3) The output of one state is used as the input to another state when there is a transition joining the two.

4) Backtracking must keep track of both previous endpoints and previous states.

Such an algorithm requires $|\Delta|$ time warps—i.e., one per transition, where $|\Delta| \leqslant |V| \cdot |S|$ since $\Delta(s, q)$ need not be defined for all s and q. The algorithm also requires $4|S|M$ storage locations—i.e., one for each of the accumulated distances, the associated word, the associated previous ending point, and the associated previous state.

VI. COMPUTATIONAL COMPARISONS OF CONNECTED WORD DTW ALGORITHMS

It is worthwhile comparing the level building DTW algorithm to the two-level DP warp of Sakoe [10], and the sampling DTW algorithm of Rabiner and Schmidt [11]. The comparisons will be in terms of both computation and storage for processing an L-digit string.

A. Computation

For the level building algorithm with local and global range constraints of 2 to 1, and $\frac{1}{2}$ to 1 (i.e., the range falls within a parallelogram of the type shown in Fig. 7), the number of local distance calculations of Algorithm 5 is

$$ND_{A5} = V \cdot L_{\mathrm{MAX}} \cdot \bar{N} \cdot M/3 \tag{48}$$

where $\bar{N}$ is the average duration (in frames) of a reference pattern, L_{MAX} is the number of levels (if $L < L_{\mathrm{MAX}}$, then generally L_{MAX} levels will not be required), V is the number of references per level, M is the duration of the test pattern, and $\bar{N} \cdot M/3$ is the average number of distances at each level.

For the two-level DP warp of Sakoe, the number of local distance calculations is

$$ND_{S_1} = V \cdot M \cdot \bar{N} \cdot (2R + 1) \tag{49}$$

where R is a range parameter for the fixed range DTW algorithm used in this method [5].

For the sampling DTW algorithm of Rabiner and Schmidt [11], the number of local distance calculations is

TABLE I-B
TYPICAL COMPUTATIONAL REQUIREMENTS FOR THE CASE $L_{MAX} = 5$, $V = 10$, $M = 120$, $\bar{N} = 35$, $\bar{\epsilon} = 8$, $\epsilon = 12$, $R = 12$, $\bar{\gamma} = 1.5$, $L = 4$

DTW Algorithm

	Level Building	Two-Level DP Warp	Sampling	Reduced Level Building
Number of Basic Time Warps	50	1200	60	40
Size of Time Warps	1400	875	595	875
Total Computation For Distances	70,000	1,050,000	35,700	35,000
Storage	1800	6000	0	1800

$$ND_{RS} = V \cdot L \cdot \bar{N} \cdot (2\bar{\epsilon} + 1) \cdot \bar{\gamma} \tag{50}$$

where $\bar{\epsilon}$ is a range parameter for a local minimum DTW algorithm [7], and $\bar{\gamma}$ represents the average number of candidate strings being processed. (Rabiner and Schmidt [11] showed that on average $1 \leqslant \bar{\gamma} \leqslant 2$.)

Finally, by incorporating the range reduction techniques of Section V into the algorithm, the number of local distance calculations of the "reduced level building" DTW algorithm becomes

$$ND_{A5R} = V \cdot L \cdot \bar{N} \cdot (2\epsilon + 1) \tag{51}$$

where ϵ is the range parameter for following the local minimum.

Table I-A summarizes the computational aspects of these connected word DTW algorithms. The row labeled number of basic time warps refers to the number of times the DTW algorithm is applied. The size of the time warp is the (average) size of the region in the (m, n) plane covered by a typical time warp, and the total is the number of distances of (48)-(51). (The row labeled storage will be explained below.)

Table I-B gives a numerical comparison of the computation for the typical parameter values

$$L_{\max} = 5, \quad V = 10, \quad M = 120, \quad \bar{N} = 35, \quad \bar{\epsilon} = 8, \quad \epsilon = 12,$$
$$R = 12, \quad \bar{\gamma} = 1.5, \quad L = 4.$$

It can be seen that the two-level DP Warp of Sakoe needs to do about 20 times more time warps than any of the other algorithms, and that, for this example, the ratio between the total computation of the two-level DP Warp method, and the reduced level building algorithm is 30 to 1! It can also be seen in Table I-B that the total computation is comparable for the sampling and reduced level building algorithms.

B. Storage

The storage requirements for the different connected word DTW algorithms refer to storage for arrays that are specific to the algorithm—hence, storage for reference patterns, and accumulated distance functions ($D_l(m, n)$), which are common to all algorithms, are not included here. The storage of the level building algorithm is that required for the vectors $\tilde{D}_l^B(m)$, $W_l(m)$, $\tilde{F}_l^B(m)$. Each of these vectors is of size M. Hence, the total storage of both the level building, and the reduced level building algorithm is

$$S_{A5} = S_{A5R} = 3 \cdot M \cdot L_{\text{MAX}}. \tag{52}$$

The storage for the two-level DP matching algorithm is

$$S_{S1} = 2 \cdot M \cdot (2R + 1) \tag{53}$$

for the matrices of distance and best word, for each of the $(2R + 1)$ pairs of beginning and ending points. The storage for the sampling algorithm is essentially 0 since only a small list (125 locations) of accumulated strings is retained in this method.

The storage requirements of the four algorithms are summarized in the last row of Table I-A, and some numerical comparisons are given in the last row of Table I-B. It can be seen that the two-level DP warp method requires about three times the storage of the level building algorithm.

VII. Summary

A novel approach to dynamic time warping alignment of a connected word string and a concatenated set of reference patterns has been described. The algorithm was shown to be applicable to connected word recognition problems and a highly efficient implementation was described. A number of assumptions were used in deriving the algorithm, namely:

1) An overall assymetric function was used—not a distance per pattern.

2) Reference patterns were concatenated, not overlapped.

3) Consecutive reference patterns were independent—i.e., there was no level interaction.

The final output of the algorithm was the best concatenated reference pattern that matched the input string.

A number of modifications to the algorithm were described to increase the flexibility and accuracy of the method. In a companion paper we present results on the use of the reduced level building algorithm to a connected digit recognition application. There it is shown that a highly accurate and efficient connected digit recognizer can be implemented using the level building algorithms.

Appendix

Theorem: If there exists a super reference pattern $R^s_{r_1 r_2 \cdots r_l}$ such that

$$D_{r_1 r_2 \cdots r_l}(e_l) \le D_{s_1 s_2 \cdots s_l}(e_l)$$

for any $R^s_{s_1 s_2 \cdots s_l}$, then $D_{r_1 r_2 \cdots r_l r_{l+1}}(e_{l+1}) \le D_{s_1 s_2 \cdots s_l r_{l+1}} \cdot (e_{l+1})$ for any $R_{s_1 s_2 \cdots s_l}$ and any $R_{r_{l+1}}$.

Proof: Let $R^s_{s_1 s_2 \cdots s_l}$ be a general super reference pattern of length l and let $R_{r_1 r_2 \cdots r_l}$ be the super reference pattern which minimizes the distance $D_{s_1 s_2 \cdots s_l}(e_l)$ over all possible $s_1 s_2 \cdots s_l$. Then, by definition, we have

$$D_{s_1 s_2 \cdots s_l}(e_l) = \sum_{m=1}^{e_l} d(m, w(m)) \tag{A1}$$

and

$$D_{s_1 s_2 \cdots s_l r_{l+1}}(e_{l+1}) = \sum_{m=1}^{e_{l+1}} d(m, w(m)) \tag{A2}$$

$$= \sum_{m=1}^{e_l} d(m, w(m)) + \sum_{m=e_l+1}^{e_{l+1}} d(m, w(m)) \tag{A3}$$

$$= D_{s_1 s_2 \cdots s_l}(e_l) + \sum_{m=e_l+1}^{e_{l+1}} d(m, w(m)). \tag{A4}$$

Since the second term in (A4) is dependent only on $R_{r_{l+1}}$ and since the sequence $R_{r_1} \oplus R_{r_2} \oplus \cdots \oplus R_{r_l}$ minimizes $D_{s_1 s_2 \cdots s_l}(e_l)$, then to minimize $D_{s_1 s_2 \cdots s_l r_{l+1}}(e_{l+1})$ the best $(l + 1)$ long sequence will have $R_{r_1} \oplus R_{r_2} \oplus \cdots \oplus R_{r_l}$ as its first l reference patterns.

References

[1] H. Sakoe and S. Chiba, "A dynamic programming approach to continuous speech recognition," in *Proc. 7th Int. Congress Acoust.*, Budapest, Hungary, Paper 20C-13, 1971.

[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] 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.

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

[5] 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.

[6] 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.

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

[8] L. R. Rabiner, S. C. Levinson, A. E. Rosenberg, and J. C. Wilpon, "Speaker-independent recognition of isolated words using clustering techniques," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 336-349, Aug. 1979.

[9] J. S. Bridle and M. D. Brown, "Connected word recognition using whole word templates," in *Proc. Inst. Acoust.*, Autumn Conf., 1979.

[10] H. Sakoe, "Two-level DP-matching—A dynamic programming-based pattern matching algorithm for connected word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-27, pp. 588-595, Dec. 1979.

[11] L. R. Rabiner and C. E. Schmidt, "Application of dynamic time warping to connected digit recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 377-388, Aug. 1980.

[12] 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. 623-635, Dec. 1980.

[13] L. R. Bahl and F. Jelinek, "Decoding for channels with insertions, deletions, and substitutions with applications to speech recognition," *IEEE Trans. Inform. Theory*, vol. IT-21, pp. 404-411, July 1975.

[14] C. S. Myers and L. R. Rabiner, "Connected digit recognition using a level building DTW algorithm," *IEEE Trans. Acoust., Speech, Signal Processing*, to be published.

Cory S. 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) was born in Brooklyn, NY, on September 28, 1943. He received the S.B. and S.M. degrees simultaneously in 1964, and the Ph.D. degree in electrical engineering in 1967, all from the Massachusetts Institute of Technology, Cambridge.

From 1962 through 1964, he participated in the cooperative plan in electrical engineering at Bell Laboratories, Whippany 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, NJ. He is coauthor of the books *Theory and Application of Digital Signal Processing* (Englewood Cliffs, NJ: Prentice-Hall, 1975) and *Digital Processing of Speech Signals* (Englewood Cliffs, NJ: 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 past President of the IEEE Acoustics, Speech, and Signal Processing Society Ad Com, a member of the Acoustics, Speech, and Signal Processing Society Technical Committee on Digital Signal Processing, a member of the Acoustics, Speech, and Signal Processing Society Technical Committee on Speech Communications, a former Associate Editor of the IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, a member of the PROCEEDINGS OF THE IEEE Editorial Board, and a former member of the Technical Committee on Speech Communication of the Acoustical Society of America.

Direct (Nonrecursive) Relations Between Cepstrum and Predictor Coefficients

MANFRED R. SCHROEDER, FELLOW, IEEE

Abstract—Direct, i.e., nonrecursive, relations are derived for the cepstrum in terms of the predictor coefficients and vice versa. Connections with algebraic roots, symmetric functions, statistical moments, and cumulants are pointed out. Some implications for pitch detection are also discussed.

INTRODUCTION

RECURSIVE relations between cepstrum and predictor coefficients [1] have long been known [2]. For some purposes, knowledge of *direct* relations between these two sets of important parameters characterizing sources and signals is desirable.

Manuscript received October 5, 1979; revised June 13, 1980 and July 9, 1980.

The author is with Drittes Physikalisches Institut, Universitaet Goettingen, Goettingen, Germany, and Bell Laboratories, Murray Hill, NJ 07974.

DERIVATION OF THE MAIN RESULT

Let

$$A(z) = \sum_{k=0}^{p} a_k z^{-k}; \qquad a_0 = 1; \;\; a_p \neq 0 \tag{1}$$

be an "inverse filter" polynomial [2] of order p whose roots are inside the unit circle. The a_k are the predictor coefficients. Then $1/A(z)$ is a (stable) all-pole filter whose cepstrum coefficients c_n are customarily defined by

$$ln[1/A(z)] =: \sum_{n=1}^{\infty} c_n z^{-n}. \tag{2}$$

The well-known recursive relation between the a_k and c_n is obtained by differentiating (2) with respect to z^{-1} and equating equal powers of z^{-1}, yielding [3]