Two-Level DP-Matching—A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition

HIROAKI SAKOE

Abstract—**This paper reports a pattern matching approach to connected word recognition. First, a general principle of connected word recognition is given based on pattern matching between unknown continuous speech and artificially synthesized connected reference patterns. Time-normalization capability is allowed by use of dynamic programming-based time-warping technique (DP-matching). Then, it is shown that the matching process is efficiently carried out by breaking it down into two steps. The derived algorithm is extensively subjected to recognition experiments.**

It is shown in a talker-adapted recognition experiment that digit data (one to four digits) connectedly spoken by five persons are recognized with as high as 99.6 percent accuracy. Computation time and memory requirement are both proved to be within reasonable limits.

I. Introduction

CONNECTED word recognition is one of the most interesting problems at the present stage of speech recognition research since isolated word recognition techniques have been improved up to a practically high level [4], [8]. There are two reasons why this problem is so focused. First of all, connected speech input improves data input rate, as well as recognition system operational facility. Numeric data input, involving pronouncing numbers on a digit-by-digit basis, would be unacceptably slow and irritating work. Apart from these practical aspects, connected word recognition research should be emphasized from more long-term viewpoints, as well. This research will result in some key techniques for natural continuous speech recognition.

The target of this investigation is a talker-dependent high-accuracy recognition system, which deals with word-connected speech involving a limited vocabulary. The basic recognition strategy employed is the so-called pattern matching method. Though it is simple in principle, it is a very effective and efficient recognition method, as long as it is performed on a word unit basis.

Recent dynamic programming-based time-normalization technique (DP-matching [1]-[4]) has greatly improved its practical effectiveness. The pattern matching method also possesses many convenient features. It can be applied to different situations without any recognition algorithm modification. Vocabulary change or system adaptation to individual speakers is easily performed by updating the reference pattern. Furthermore the pattern matching method works fairly well for any foreign language. In this paper pattern matching is carried out between speech patterns with quasi-continuous time axes. Here, the term "quasi-continuous" means that time sampling is made with a relatively short (10 ms or so) constant period. It is well known that the resulting speech pattern is then very redundant. From the computational economy standpoint, quasi-continuous sampling will be insufficient. Some phonemic level segmention is naturally expected to reduce the redundancy. In spite of these well-known factors, preliminary segmentation is utterly discarded in this investigation. It is considered that computational efficiency improvement is a secondary problem, coming after the establishment of a more reliable recognition principle.

Two-level DP-matching discussed in this paper is a connected word recognition method based on pattern matching. It employs no preliminary segmentation in principle. Pattern matching is made on a whole connected word basis, that is, between unknown continuous and artificially synthesized connected reference patterns. The time-normalizing feature is given through the use of dynamic programming. It is shown that the matching process is efficiently carried out by breaking it down into two steps, word level matching and phrase level matching. Two execution algorithms are derived which are suitably applicable to the computer simulation experiment and the real-time recognition system, respectively. High-level recognition performance was established through several connected word recognition experiments. Computation time and memory requirements are both estimated to be within reasonable limits.

II. Connected Word Recognition Principle

A. Preliminary Discussions

As the result of the discussions in Section I, it has been decided that recognition should be made based on pattern matching with word unit reference patterns. In such a simplified case, the principal problems are considered to be segmentation, time-normalization, and coarticulation problems.

Segmentation is now a word level problem. In other words, this segmentation means separating a continuous speech into word units. This word level segmentation seems much easier than the phoneme or syllable level segmentation. Actually, Sambur and Rabiner [9] have reported a segmentation rule applied to English digit words. Their rule, however, was

Manuscript received August 11, 1978; revised April 10, 1979 and July 23, 1979.

The author is with Central Research Laboratories, Nippon Electric Company, Kawasaki, Japan.

0096-3518/79/1200-0588$00.75

specially designed for a specific vocabulary. It seems difficult to generalize their rule to a point where it would be applicable to an arbitrarily selected word set recognition. As vocabulary size grows, the rule will become more complicated. As long as the segmentation is imperfect, highly complicated recognition rules will be required to compensate for the imperfection. These considerations suggest the desirability of not making any segmentation prior to recognition, as was proposed by Sakoe and Chiba [2], [3]. In other words, the word boundary decision should be made in parallel with the word category decision.

Time-normalization is one of the central problems, even in isolated word recognition research. There should be much more time-normalization effort made in connected word recognition because time-axis fluctuation is much more violent in the continuous speech context than in the isolated speech context. Fortunately, dynamic programming-based pattern-matching technique, or DP-matching, has almost completely solved the problem [1]-[4]. In the early stage of DP-matching evolution a trial application is made to the connected word recognition problem [2], [3]. It has been reported that DP-matching is one of the most promising approaches to connected word recognition.

In the present study, no active effort is made to normalize or avoid the coarticulation effect. Instead, it is expected that in the case of word-unit-based recognition, the coarticulation effect is less severe than in the case of phoneme-unit-based recognition. Besides, multiple reference pattern use will solve the problem to a certain practical extent.

B. Time-Normalization Based on Dynamic Programming

In the proposed connected word recognition algorithm dynamic programming, or DP-matching, plays an essential role. A summarized description is given here on the DP-matching algorithm. Let

$$A = \mathbb{a}_1, \mathbb{a}_2, \cdots, \mathbb{a}_i, \cdots, \mathbb{a}_I$$
$$B = \mathbb{b}_1, \mathbb{b}_2, \cdots, \mathbb{b}_j, \cdots, \mathbb{b}_J \tag{1}$$

be two speech patterns. Each of them is represented as a sequence of feature vectors with quasi-continuous time axis i or j. The time-normalized distance measure between these two speech patterns is defined as

$$D(A,B) = \min_{j=j(i)} \left[\sum_{i=1}^{I} d(i,j) \right] \tag{2}$$

where $d(i,j)$, vector distance, is the distance between vectors $\mathbb{a}_i$ and $\mathbb{b}_j$. Warping function $j(i)$ is subjected to monotonic condition, continuous condition, etc., so that it can well approximate the actual time-axis fluctuation property. (See Sakoe and Chiba [4].) The minimization problem (2) is very efficiently solved by the use of dynamic programming.

The above definition is essentially the same as the asymmetric form distance given by Sakoe and Chiba [4]. It has been reported that the symmetric form distance shows significantly better performance than the asymmetric one. The reason why the asymmetric form distance is employed, in spite of its relative inferiority, will be made clear through the discussion in Section II-D.

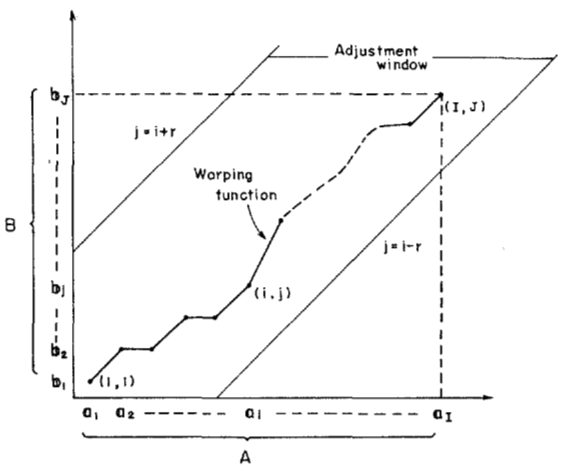


Fig. 1. DP-matching principle.

In order to give a basis for computational efficiency estimation in the following discussions, the computation time consumed in a DP-matching process is analyzed here. The DP-matching algorithm is intuitively well explained by the i-j plane shown in the Fig. 1. The asymmetric form DP-equation [4]

$$g(i,j) = \min \begin{bmatrix} g(i-1, j-2) + (d(i, j-1) + d(i,j))/2 \\ g(i-1, j-1) + d(i,j) \\ g(i-2, j-1) + d(i-1, j) + d(i,j) \end{bmatrix} \tag{3}$$

is recurrently calculated in the adjustment window $|i-j| \leqq r$ starting from point $(1,1)$ up to (I,J). Therefore, computation time consumed by one DP-matching process is nearly proportional to $(2r+1) \times J$. This product is used as the basis for algorithm efficiency evaluation.

C. Connected Word Recognition Principle

Let $1, 2, \cdots, n, \cdots, N$ represent an N word vocabulary. The reference pattern of word n is represented as

$$B^n = \mathbb{b}_1^n, \mathbb{b}_2^n, \cdots, \mathbb{b}_j^n, \cdots, \mathbb{b}_{J^n}^n. \tag{4}$$

Let

$$C = \mathbb{c}_1, \mathbb{c}_2, \cdots, \mathbb{c}_i, \cdots, \mathbb{c}_I \tag{5}$$

be an unknown input speech pattern. This input speech may be a single word speech or a multiple word speech. Hereafter, it is called connected speech or a phrase. The basic concept of the proposed recognition principle is shown in Fig. 2. An operator "$\oplus$" is employed (meaning concatenating two speech patterns) as, for example,

$$B^m \oplus B^n = \mathbb{b}_1^m, \mathbb{b}_2^m, \cdots, \mathbb{b}_{J^m}^m, \mathbb{b}_1^n, \mathbb{b}_2^n, \cdots, \mathbb{b}_{J^n}^n. \tag{6}$$

An approximated phrase reference pattern $\overline{B}$ of words $n(1)$, $n(2), \cdots, n(k)$ is synthesized by concatenating their reference patterns as

$$\overline{B} = B^{n(1)} \oplus B^{n(2)} \oplus \cdots \oplus B^{n(k)}.$$

Pattern matching is made between unknown input pattern C and above synthesized reference pattern $\overline{B}$, providing a

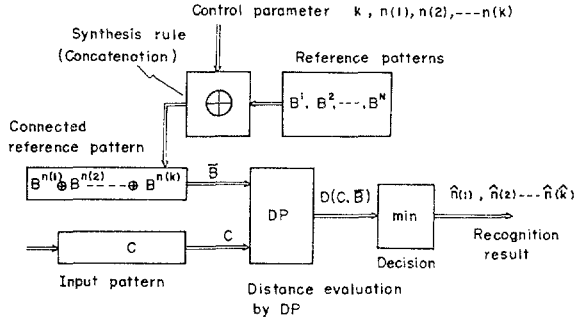


Fig. 2. Proposed connected word recognition principle.

distance $D(C, \bar{B})$. These processes are repeated, changing the number of words k and indexes $n(1), n(2), \cdots, n(k)$. When the above process is carried out until all repeated permutations of the indexes are exhausted, optimum parameters $k = \hat{k}$ and $n(x) = \hat{n}(x)$, $x = 1, 2, \cdots \hat{k}$ are determined, which give minimum distance $D(C, \bar{B})$. Then, decision is made that input pattern C comprises $\hat{k}$ words $\hat{n}(1), \hat{n}(2), \cdots, \hat{n}(\hat{k})$. Mathematically, this principle is formulated in the following minimization problem format.

$$T = \min_{k, n(x)} [D(C, B^{n(1)} \oplus B^{n(2)} \oplus \cdots \oplus B^{n(x)} \oplus \cdots \oplus B^{n(k)})]. \tag{8}$$

This recognition scheme needs no preliminary segmentation because pattern matching is made on a whole phrase basis. Thus, erroneous recognition possibility caused by inaccurate segmentation is completely excluded. Accordingly, a significantly higher performance is expected compared with traditional preliminary segmentation making approaches.

The minimization problem (8), however, is not an easy problem to be solved by the so-called direct search, or exhaustive comparison, method. It consumes a prohibitively enormous amount of computation because all possible concatenations of the reference patterns must be tested.

Sakoe and Chiba [2], [3] reported an approximate solution method for this problem. Their method is called the previous method in the following part of this paper. It worked fairly well for two-digit connected speech. Nevertheless, when word connection was further increased, recognition accuracy became rapidly degraded. It should be noted, however, that they made use of an interesting property of dynamic programming which gives solutions for a set of different boundary conditions in parallel.

On the premise that the syllable level (VCV-unit) segmentation is made on a preliminary stage, Nakatsu and Kohda [5] gave a solution method for problem (8). It is a complete and efficient method, as long as it is applied after accurate segmentation is made. On the other hand, the present investigation is on a quasi-continuous time axis basis, as declared in Section I of this paper. The following sections report an efficient segmentation-free recognition algorithm based on the efforts reported by Sakoe and Chiba and by Nakatsu and Kohda.

D. Minimization Process Decomposition

In this section the minimization problem (8) is broken down into two steps: one for word unit level, and one for the whole

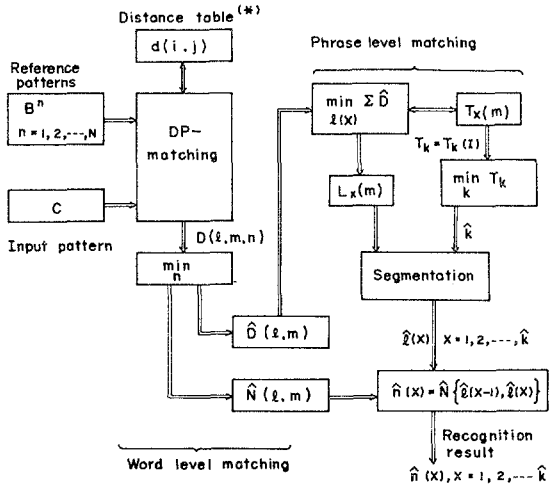


Fig. 3. Schematic diagram of two-level DP-matching. (*) Vector distance table is used only in Algorithm (I).

connected speech level (or phrase level). A partial pattern $C(l, m)$ for input pattern C is defined as follows:

$$C(l, m) = \mathbb{c}_{l+1}, \mathbb{c}_{l+2}, \cdots, \mathbb{c}_m. \tag{9}$$

$(k - 1)$ word boundary timings $l(1), l(2), \cdots, l(k - 1)$ are assumed on the input pattern C time axis while breaking down input pattern C into k partial patterns

$$C = C(l(0), l(1)) \oplus C(l(1), l(2)) \oplus \cdots \oplus C(l(k - 1), l(k)) \tag{10}$$

where $l(0) = 0$ and $l(k) = I$. Asymmetric form distance (2) holds the following property in relation to the speech pattern breakdown:

$$D(C, B^m \oplus B^n) = \min_{l} [D(C(0, l), B^m) + D(C(l, I), B^n)]. \tag{11}$$

By putting (10) into (8), and repeatedly applying the relation in (11), we obtain

$$T = \min_{k, l(x)} \left[\sum_{x=1}^{k} \min_{n(x)} [D(l(x - 1), l(x), n(x))] \right] \tag{12}$$

where notation $D(l, m, n)$ is an abbreviation of $D(C(l, m), B^n)$, that is, a distance between partial pattern $C(l, m)$ and reference pattern B^n. Note that relation (11) does not hold for the symmetric form distance defined in [4]. Accordingly, (12) will not result. This is the reason why the asymmetric form distance definition (2) is adopted, in spite of its performance inferiority to the symmetric form distance definition.

There are two minimization problems involved in (12). They are efficiently solved by the following two step algorithm. Fig. 3 explains the algorithm.

1) Word Level Matching Process: Calculate and memorize. *Partial distance:*

$$\hat{D}(l, m) = \min_{n} [D(l, m, n)]. \tag{13}$$

Partial decision:

$$\hat{N}(l, m) = \underset{n}{\operatorname{argmin}} [D(l, m, n)] \tag{14}$$

for every combination of l and m, where $0 \leqq l < m \leqq I$. (Operator "argmin" means to find optimum parameter n.)

2) Phrase Level Matching Process: Solve the minimization problems

$$T_k = \min_{l(x)} \left[\sum_{x=1}^{k} \hat{D}(l(x-1), l(x)) \right] \tag{15}$$

and then,

$$T = \min_{k} [T_k] \tag{16}$$

yielding optimum parameters $k = \hat{k}$ and $l(x) = \hat{l}(x)$, where $x = 1, 2, \cdots, \hat{k}$.

Thus, the minimization problem (8) is broken down into two steps. Resulting optimum parameters $\hat{k}$ and $\hat{l}(x)$, with the above memorized partial decisions $\hat{N}(l, m)$, give the following recognition result.

3) Decision Making Process:

$$\hat{n}(x) = \hat{N}(\hat{l}(x-1), \hat{l}(x)), \tag{17}$$

where

$$x = 1, 2, \cdots, \hat{k}.$$

Algorithm details are given in the following section, where it is shown that both of the above two steps are very efficiently carried out by use of the dynamic programming algorithm. This is the reason why the proposed algorithm is called two-level DP-matching.

III. Practical Two-Level DP-Matching Algorithm

A. Basic Operations

In the word level matching process, (13) and (14) are calculated. The process is divided into two operations: DP-matching computation between partial pattern $C(l, m)$ and reference pattern B^n, and simple comparison of resulting distances $D(l, m, n)$, $n = 1, 2, \cdots, N$. A large amount of the computations is occupied by the DP-matching operation. Distances $D(l, m, n)$ must be calculated for every possible combination of l, m, and n. Fortunately, dynamic programming inherently has a convenient property which simultaneously furnishes solutions for different boundary conditions. This is well explained by the following concrete algorithm description and the i-j plane in Fig. 4. DP-matching for partial patterns, $C(l, m)$ beginning at $i = l + 1$, is executed by a DP-equation (3) calculation with the initial condition $g(l+1, 1) = d(l+1, 1)$ within the adjustment window $l + j - r \leqq i \leqq l + j + r$ up to $j = J^n$. When the above process is completed, we obtain $(2r+1)$ distances $D(l, m, n) = g(m, J^n)$ simultaneously within the adjustment window at $j = J^n$

$$l + J^n - r \leqq m \leqq l + J^n + r. \tag{18}$$

Originally, adjustment window length was set so that it can cover the actual timing difference between reference pattern and input pattern. There is no need to take into account those

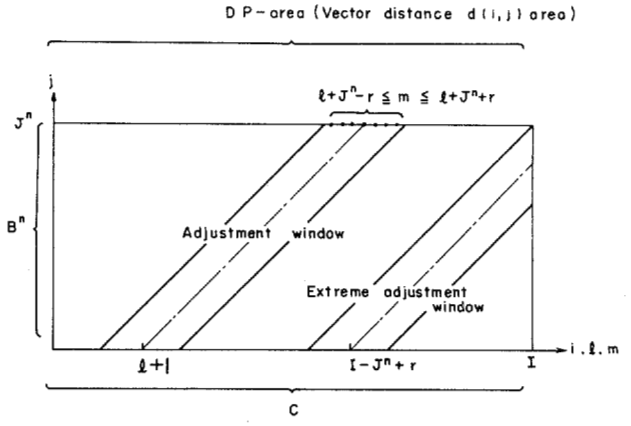


Fig. 4. Word level matching in Algorithm (I). Vector distances are calculated in $1 \leqq i \leqq I$ and $1 \leqq j \leqq J^n$. DP-matching is repeated scanning $0 \leqq l \leqq I - J^n + r$.

partial patterns whose ending points m are separated from the adjustment window (18). Thus, it has been shown that one DP-matching operation gives all necessary distances $D(l, m, n)$ for a current (l, n) value. The above DP-matching property cuts down computation time considerably. The variable l domain is, as is seen from Fig. 4,

$$0 \leqq l \leqq I - J^n + r. \tag{19}$$

Of course, $1 \leqq n \leqq N$.

In the phrase level matching process, minimization problem (15) is solved, again using the dynamic programming algorithm

Initial condition: $T_0(0) = 0$

DP-equation: $$T_x(m) = \min_{l} [\hat{D}(l, m) + T_{x-1}(l)], \tag{20}$$

where $m = 1, 2, \cdots, I$, and $x = 1, 2, \cdots, \bar{k}$. ($\bar{k}$ is a prespecified maximum word number.) Along with the above DP-equation calculation, optimum parameter

$$L_x(m) = \underset{l}{\operatorname{argmin}} [\hat{D}(l, m) + T_{x-1}(l)] \tag{21}$$

is determined and memorized to form an $L_x(m)$ table.

After completion of the above process, T_k is given by $T_k = T_k(I)$, where $k = 1, 2, \cdots, \bar{k}$. Optimum word number $\hat{k}$ becomes definite, as

$$\hat{k} = \underset{k}{\operatorname{argmin}} [T_k]. \tag{22}$$

(Operator "argmin" means to find optimum parameter k.)

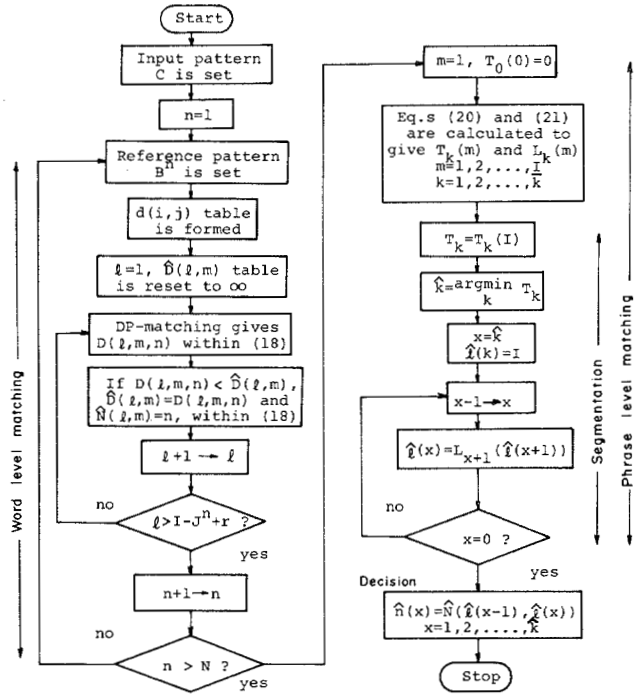
Then, optimum word boundaries $\hat{l}(x)$ are decided by refering to the $L_x(m)$ table as

$$\hat{l}(\hat{k}) = I$$

$$\hat{l}(x) = L_{x+1}[\hat{l}(x+1)], \; x = \hat{k} - 1, \hat{k} - 2, \cdots, 1. \tag{23}$$

Decision is made based on the above resulting optimum word boundaries $\hat{l}(1), \hat{l}(2), \cdots, \hat{l}(k)$, as is shown by (17).

The above described basic operations are executable in different computational ways. Two typical versions of practical algorithms are described in the following. One is a computer simulation suitable algorithm. Instead of occupying relatively



Note - Operator "$\underset{k}{\text{argmin}}$" means to find optimum parameter k.

Fig. 5. Algorithm (I) flowchart.

large working memories, it performs a two-level DP-matching process with minimum computation effort. The other algorithm is devised for a real-time recognition system. Although this method is rather complicated, much of its operations can be performed in parallel with speech pattern input. Besides, the working memory requirement is quite reasonable.

B. Algorithm (I)

In general, much of the computation for two-level DP-matching is consumed in the word level process, in particular in the DP-equation (3) calculation. In more detail, the calculation of vector distance $d(i,j) = \|c_i - b_j\|$ in the DP-equation (3) takes up a considerable part of the computations because both vectors $\mathbb{c}_i$ and $\mathbb{b}_j$ are usually more than ten-dimensional. Therefore, duplicate calculations of the same distances $d(i, j)$ should be avoided for any combination of i and j. In the case of nonreal-time recognition, where the recognition process starts after an input speech pattern C has been completely fed into the system, all necessary vector distances $d(i, j)$ can be calculated to form a distance table. The table is two dimensionally organized so that it covers the $I \times J^n$ square shown in Fig. 4. Algorithm details are shown in the Fig. 5 flowchart. The schematic diagram is roughly the same as that of Fig. 3. For the DP-equation (3) calculation, vector distances $d(i, j)$ are provided from this distance table. Therefore, duplicate vector distance calculation does not occur, in spite of the wide intersection of directly consecutive DP-matching adjustment windows $l + j - r \leqq i \leqq l + j + r$ and $l + j - r + 1 \leqq i \leqq l + j + r + 1$. DP-matching with a reference pattern is repeatedly preformed with its adjustment window incrementally shifted, along with partial pattern beginning timing l. As DP-matching has completely scanned the Fig. 4 distance table, or more definitely, when partial pattern beginning point l is incrementally changed to exhaust the region (19), the table is refreshed for the next reference pattern.

TABLE I
TYPICAL DIMENSIONS IN FIVE DIGIT NUMERICAL RECOGNITION CASE*

Parameter (unit)	Exhaustive Comparison	Two Level DP
Input pattern length I (frame)	100	100
Reference pattern length J (frame)	100 (5-digit connected)	20 (per digit average)
J_{max} (frame)	-	25
J_{min} (frame)	-	15
Vocabulary size N (word)	10	10
Number of words connected K (word)	5	5
Window length (frame)	40 (=0.4xJ)	8 (=0.4xJ)

* Speech pattern sampling period is assumed to be 18 ms.

In this algorithm version phrase level matching begins after the word level matching process is accomplished. Therefore, complete sets of partial distances $\hat{D}(l, m)$ and partial decisions $\hat{N}(l, m)$ must be memorized so that they can be accessed during the phrase level matching and decision making processes. Address l, or partial pattern beginning timing, takes values within (19). Address m, or partial pattern ending timing, depending on l value, takes value within (18) for each reference pattern B^n. Therefore, address m must cover their union

$$l + J_{min} - r \leqq m \leqq l + J_{max} + r \tag{24}$$

where $J_{min} = \min_n [J^n]$ and $J_{max} = \max_n [J^n]$. Referring to Table I, a numerical example of an actual case, it can be seen that the memory requirement for these two-dimensionally organized tables is of some considerable size (2900 words for each table, which may further grow in proportion to input pattern length I). In addition, a distance table and input pattern buffer occupy a sizable memory area. However, these memory requirements are not so unreasonable when the algorithm is coded on a general purpose computer. Therefore, it can be said that this algorithm is especially suitable to computer-simulated experiment application.

With relation to the phrase level matching process, there is little to supplement. When (20) and (21) are calculated, parameter l should be in

$$m - J_{max} - r \leqq l \leqq m - J_{min} + r, \tag{25}$$

which comes from the condition in (24). Obviously, those partial distances $\hat{D}(l, m)$, for which $l < 0$, should be counted out. Or, more precisely, parameter l should be within $\max [0, m - J_{max} - r] \leqq l \leqq m - J_{min} + r$.

C. Algorithm (II)

It is obvious that a DP-matching process can be executed in a time-reversed fashion. The initial condition is given at (m, J^n). Time-reversed DP-equation (which is obtained from (3) by substituting $i + 1, i + 2, j + 1$ and $j + 2$ for $i - 1, i - 2, j - 1$ and $j - 2$, respectively) is recurrently computed in descending order with relation to i and j. Then, distance $D(l, m, n) =$

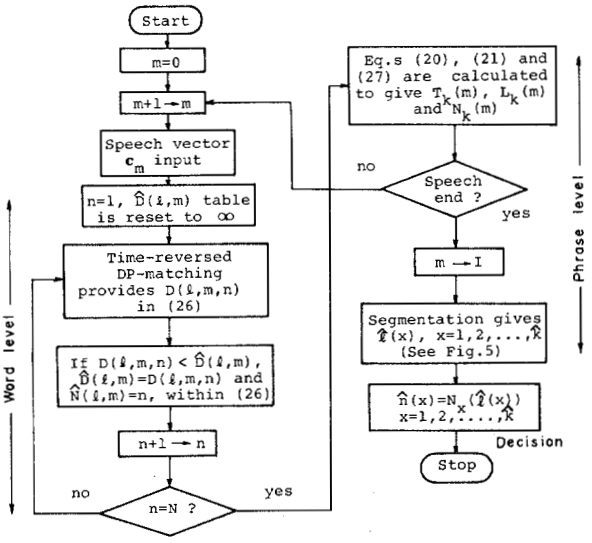


Fig. 6. Algorithm (II) flowchart.

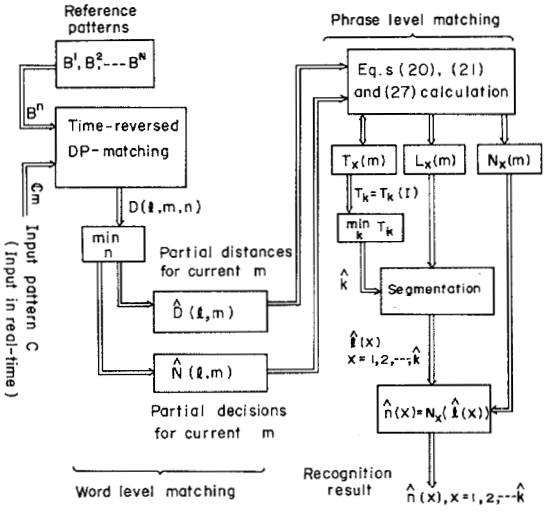


Fig. 7. Schematic diagram of Algorithm (II).

$g(l+1, 1)$ is yielded for each l within

$$m - J^n - r \leqq l \leqq m - J^n + r. \tag{26}$$

When the above DP-matching processes have been repeatedly performed for all reference patterns B^n, partial distances $\hat{D}(l, m)$ and partial decisions $\hat{N}(l, m)$ are completely furnished within (25) for a current m value. In phrase level matching, $T_x(m)$ in (20) can be computed as soon as all partial distances $\hat{D}(l, m)$, for which l values are within (25), are prepared.

These considerations lead to a real-time processing algorithm, shown in the Fig. 6 flowchart and Fig. 7 schematic diagram. Word level and phrase level matching operations proceed in parallel, synchronized with input pattern vector c_m feed-in-timing. Partial decisions $\hat{N}(l, m)$, also, are subjected to processing along with (20) calculation, as follows:

$$N_x(m) = \hat{N}(\hat{l}, m) \tag{27}$$

where $\hat{l}(=L_x(m))$ is the optimum l value which attains (20) minimization.

Buffer memories for $\hat{D}(l, m)$ and $\hat{N}(l, m)$, for which address l is within (25), can be refreshed every time new input vector C_m is fed in. Therefore, each of them now is of one-dimensioal organization, and occupies only $(J_{max} - J_{min} + 2r + 1)$ words [this comes from (25)]. Besides, there is no need for buffering whole input pattern C nor whole vector distances $d(i, j)$.

In this algorithm a word level matching process, which intrinsically consumes many computations, is embedded into the input utterance duration. Therefore, a relatively large amount of computation requirement, caused by duplicate calculation of vector distance $d(i, j)$, does not matter so much in practice. This real-time processing feature, as well as the smallest memory requirement, implies that this algorithm is the most promising one for real-time system implementation.

D. Efficiency Evaluation

Here, the proposed algorithm efficiency is quantitatively analyzed in comparison with the case where the minimization problem in (8) is solved by Fig. 2's exhaustive comparison method. The numerical example in Table I is used as a typical case. In the exhaustive comparison method, $N^k = 10^5$ connected reference patterns are synthesized to be subjected to the DP-matching operation. It is assumed that the number of words k in the input pattern is known to be 5. The adjustment window length r is set so that it covers a ±40 percent timing difference, or $r = 0.4J$. Therefore, as was described in Section II-B, approximately 8.1×10^3 times DP-equation computations are consumed for each synthesized reference pattern. Thus, total DP-equation computation is estimated to amount to 8.1×10^8. In the proposed method, about $I \times N$ times DP-matching process must be repeated. Reference patterns are now of word units ($J = 20$). Therefore, it is estimated that total DP-equation iterations are about 3.4×10^5 times. Thus, it is known that the efficiency of the proposed method is more than 10^3 times as high as that of the exhaustive comparison method.

The above evaluation is made using the comparison scale of DP-equation iteration times. It is obvious that Algorithm (I) is much faster than the above estimation because duplicate vector distance calculation is omitted. In the Algorithm (II) case, the above estimation holds as long as the computation amount is concerned. However, it should be noted that, in Algorithm (II), much of its computations can be performed in parallel with real-time speech pattern input.

There are many varieties of practical algorithms differing from the ones described above. However, it seems that those which employ less working memories take more computations. The two above algorithms appear to be the most suitable ones for the computer simulation system and for a real-time system, respectively.

E. Word Number Specification Technique

It is feared, in connected word recognition, that word number decision errors might eventually occur. Here, "word number decision errors" means such case where two-word connected speech is recognized as three words, for example. Differing from substitution error, this is a problem peculiar to

connected word recognition. Where input format (word connection number) is prespecified, there is a way to eliminate or diminish this type of error. Let word connection number k be specified as $k \in K$, where K is a set of integers. Parameter k in (22) is subjected to condition $k \in K$. When the word connection number is uniquely specified, as $K = \{3\}$ for example, word number decision error can be completely excluded. If $K = \{1, 3, 5\}$ or $K = \{2, 4, 6\}$ for example, then much of the word number decision errors will be prevented. At least, insertion or deletion of one word will be eliminated.

IV. Experiments and Results

A. Experiment Outline

In order to quantitatively evaluate the two-level DP-matching algorithm, several recognition experiments were conducted. The experimental system used was quite the same as that used in the previous report [4] (NEAC-3100 computer and vocoder type spectrum analyzer), except that a 16-channel spectrum analyzer was used instead of a 10-channel one. Experiments were conducted in four parts. In the first three parts, Japanese digit words (See Sakoe and Chiba [4]) were used as the test vocabulary. Digit words, though being few in number, are one of the most difficult word sets to recognize, when connectedly spoken. Each of them is relatively short in duration. Accordingly, they suffer heavy distortions from coarticulation. In the final part of the experiments, a 50 geographical name vocabulary was tested.

The recognition algorithm used in these experiments was Algorithm (I). It occupied about 20 kW memories, including those for the working area, and not including those for reference pattern storage. The word level process was coded by assembler language, and the phrase level and decision making processes were coded by Fortran language. Adjustment window length r was set equal to 8 (which covered the utmost ±144 ms timing difference) by some preliminary experiments. Reference patterns were adapted for each speaker. More precisely, one (in the case of geographical name recognition) or two (in the case of digit recognition) complete repetitions of isolated utterances made by each speaker were used as reference patterns.

B. Experiment (I)

The objective of this experiment was to compare the proposed method with the previous method [2], [3]. The speech data used were three-digit continuous numerals spoken by one male speaker. 200 utterances, which include a total of 600 digits, were subjected to a recognition test. When the previous method was used, ten of them were misrecognized, or digit error rate was 10/600 = 1.7 percent. The proposed method, on the other hand, made no error. This comparison established the present algorithm superiority.

According to visual inspection, it was observed that the word boundary decision was made quite reasonably in most cases. Relatively large errors were observed when the same vowels appeared on both sides of a word boundary (for example, 2 - 1, /ni - it ʃ i/). It could be understood that accurate segmentation is very difficult in such cases. In spite of these segmentation errors, no erroneous recognition occurred. This is probably because the high time-normalization capability of the present algorithm makes it possible to recognize the abnormal partial pattern with incorrect endpoints.

TABLE II
Experiment (II) Result (For One to Four Digit Numerals Spoken by Five Persons)

Number of Digits Connected		1	2	3	4	Personal Accuracy (%)
Speaker / Error Count	A(*)	0	0	0	0	100
	B(*)	0	0	1	0	99.8
	C	0	0	0	1	99.8
	D	0	2	0	2	99.2
	E	0	0	0	5	99.0
Averaged Accuracy(%)		100	99.6	99.9	99.2	99.6

*: Well-experienced speaker

C. Experiment (II)

The objectives of this experiment were to investigate the algorithm performance in more general cases. Speech data, including up to four-digit numerals, were gathered from five male speakers. Each speaker made 50 utterances for each of one- (isolated) to four-digit connection. Recordings were made by reading a randomly ordered numeral list in a well noise-controlled room. 200 utterances, which include 500 digits, were recorded for each speaker. Thus, a total of 1000 utterances, or 2500 digits uttered by five speakers were subjected to a recognition test.

Experimental results are shown in Table II for each speaker. It is observed that connectedly-spoken numerals are recognized with as high as 99.6 percent accuracy. With relation to speaker individuality, it should be noted that no speaker made more than one percent error.

D. Experiment (III)

In this experiment the effectiveness of the word number specification technique was investigated [7]. When two persons tested a total of 800 numerical data (one- to four-digit) without word number specification, 16 recognition errors occurred, including 13 word number decision errors and three substitution errors. Then, the word number specification was employed, where specification was given by $K = \{1, 3, 5\}$ when isolated or three-digit numerals were tested, or by $K = \{2, 4\}$ otherwise. In this case, all the word number decision errors were eliminated without increasing any substitution error. Thus, it has been shown that the word number specification is a practical way of improving recognition system reliability.

D. Experiment (IV)

In the final part of the experiments nonnumerical word recognition was investigated. A 50 geographic name vocabulary appearing in [4] was used. One speaker made 200 utterances in the three-word connected manner. All of them were correctly recognized. It should be noted that such confusing word pairs as "Wakayama"-"Okayama," "Tokushima"-"Fukushima," and "Hyogo"-"Kyoto" were correctly recognized in the connected speech context.

V. Discussions

Experiment (I) results show that the present connected word recognition algorithm gives much better performance than the previous algorithm [2], [3]. The algorithm performance was extensively tested through Experiments (II), (III), and (IV). High recognition performance of the present algorithm has been established. High recognition accuracy of the algorithm should be attributed in part to the fact that preliminary segmentation is excluded from the recognition process. Word boundaries are determined in principle so that the optimum matching is attained between input pattern and connected reference patterns. This unique property, along with the high time-normalization capability, made it possible to recognize an arbitrarily selected word set with a practically high accuracy.

With relation to the computational aspect, the proposed algorithm may not be very concise or easy. In the experiments, NEAC-3100 computer (index modified addition time is 8 μs) took about ten seconds per digit. Thus, in spite of computational technique improvement throughout Section II and III, a considerable amount of computation time is still necessitated. Fortunately, recent high-speed integrated circuit devices and a pipeline processing technique have made it feasible to realize real-time recognition. Actually, a real-time recognition system has been constructed for practical use. It has been proved that it recognizes a 120 word vocabulary connected speech in real-time, based on Algorithm (II). The details of this real-time system have been reported by Tsuruta *et al.* [6].

Present investigations have been made with the target of a speaker-adaptive-type recognition system. However, the proposed algorithm is quite general, and it seems effective enough to be applied to speaker-independent (or training-free) recognition as a basic strategy. Two-level DP-matching will be able to solve a certain part of the problems which will appear along the time axis.

VI. Conclusions

A connected word recognition principle, which needs no preliminary segmentation, has been described. The principle realizes an optimum recognition in the sense that recognition is performed by attaining the best match between input pattern and artificially synthesized connected reference patterns. Two execution algorithms were derived, suitably applicable to the computer simulation system and the real-time recognition system, respectively. In these algorithms dynamic programming was effectively employed in two steps: first, to normalize intraword time-axis fluctuation, and then to make interword boundary decision. High-level recognition performance was proved by recognition experiments. Noticeable features of the algorithm are as follows:

1) The intrinsically unreliable preliminary segmentation process is completely discarded. Word boundary decision is accomplished simultaneously with word category decision.

2) Time-normalization capability is inherenthy high by use of dynamic programming-based time warping (DP-matching).

3) The algorithm is fast enough to realize a real-time system when recent high-speed digital hardware techniques are employed.

4) The algorithm is quite general in the sense that it is independent of vocabulary, language, and word connection number. High-level recognition performance comes from the above features 1) and 2).

Acknowledgment

The author wishes to thank Y. Kato and S. Chiba for their guidance and encouragement throughout this research. He also thanks S. Tsuruta for his help in conducting a certain part of the experiments. Finally, this paper has been improved by the insightful comments of anonymous referees.

References

[1] H. Sakoe and S. Chiba, "A similarity evaluation of speech patterns by dynamic programming" (in Japanese), *Dig. 1970 Nat. Meeting of Inst. Electron. Commun. Eng. Japan*, Aug. 1970.

[2] —, "A dynamic programming approach to continuous speech recognition," presented at 7th ICA, Aug. 1971, Paper 20 C 13.

[3] —, "Recognition of continuously spoken words based on time-normalization by dynamic programming," *J. Acoust. Soc. Japan*, vol. 77, Sept. 1971.

[4] —, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-26, Feb. 1978.

[5] R. Nakatsu and M. Kohda, "Computer recognition of spoken connected words based on VCV syllable unit" (in Japanese), Rep. 1974 Autumn Meeting of Acoust. Soc. Japan, Oct. 1974.

[6] S. Tsuruta, H. Sakoe, and S. Chiba, "DP-100 connected speech recognition system," presented at INTELCOM 1979, Feb. 1979.

[7] H. Sakoe, "A connected word recognition algorithm with reduced word number decision error," Rep. 1978 Spring Meeting of Acoust. Soc. Japan, May 1978.

[8] S. Chiba, M. Watri, and T. Watanabe, "A speaker-independent word recognition system," presented at 4th Int. Joint Conf. on Pattern Recognition, Nov. 1978.

[9] M. R. Sambur and L. R. Rabiner, "A statistical decision approach to the recognition of connected digits," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-24, Dec. 1976.