ARCHIVES OF ACOUSTICS
17, 2, 235–248 (1992)

A METHOD FOR CONNECTED WORD RECOGNITION

S. GROCHOLEWSKI

Institute of Computing Science
Technical University of Poznań
(60-965 Poznań, Piotrowo 3a)

In this paper, the description of the method for connected word recognition derived from the algorithm introduced by VINTSYUK is illustrated by means of a hypothetical example. Such a detailed presentation of the method should be useful from the practical point of view. The cited results of the real experiments confirm the ability of the method to perform reliable connected word recognition.

1. Introduction

Automatic speech recognition systems have initially been limited to the recognition of isolated speech consisting of a restricted set of vocabulary items. Among these early methods the so-called "global" approach, which requires that the incoming utterances be compared with template words, has been most popular. However, this approach fails when the utterance is composed of an unknown number of words and does not contain the pauses in between them.

During the early seventies a number of different solutions to this problem were considered in the USSR [10], Japan [7] and, subsequently, in the USA [5]. During the early eighties the approach introduced by VINTSYUK [10] which was unknown in the USA and Japan (his paper was not cited in [5], [7]), was adopted in a modified form by BRIDLE and BROWN [1] and NEY [6].

All the above solutions were discussed and compared in [4]. Their common feature is that they use a dynamic programming technique which serves to match optimally an unknown utterance with the "super" reference pattern [6] obtained by concatenation of single word templates.

A specific method used to find the optimally "super" reference pattern will be presented and illustrated by means of hypothetical data. This method is based on the approach introduced by VINTSYUK which was chosen because of the advantages pointed out in [4].

2. Dynamic programming in connected word recognition

At the beginning the reasons for which the dynamic programming technique is used in isolated word recognition systems [8], [3], with the global approach will be briefly reviewed.

This approach requires that the unknown utterance to be compared with each of the reference words from a lexicon. Two of many more possible time alignments between the utterance X and the reference word $W(n)$ are presented as paths f_1 and f_2 in Fig. 1. In the case of f_1 the unknown word was uttered exactly in the same manner as the corresponding reference word. In the case of f_2 the temporal relations between the parts of the words X and $W(n)$ are different. If we assume that the path f_2 is the set of points representing the optimal relation between the successive vectors in the utterance X and the reference word, and if we assign for each point (i, j) the distance $d(i, j)$ between the vectors, then the sum of distances corresponding to all points of the path gives the total distance $D(X, W(n))$ between the words X and $W(n)$.

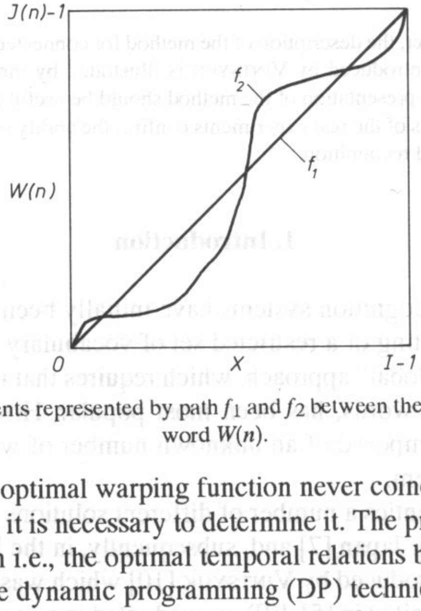


FIG. 1. Two time alignments represented by path f_1 and f_2 between the utterance X and reference word $W(n)$.

Since in practice the optimal warping function never coincides with the diagonal as in the case of f_2 in Fig. 1, it is necessary to determine it. The problem of determining the optimal warping function i.e., the optimal temporal relations between two words can be solved with the aid of the dynamic programming (DP) technique [8], [3], which allows for the assignement of the optimal path from point (0, 0) to point $(I-1, J-1)$ through the use of the following equations:

$$D(0,0) = d(0,0),$$

$$D(i,j)) = d(i,j) + \min \begin{cases} D(i-1,j) \\ D(i-1,j-1) \\ D(i,j-1) \end{cases} \tag{1}$$

where $D(i, j)$ – the minimal accumulated distance calculated from the point 0, 0 to the point i, j.

The use of the term "optimal path" signifies that the sum $D(I-1, J-1)$ of the local distances $d(i, j)$ along this path attains its minimum value.

Let us consider a more difficult case, where an utterance consists of several words uttered without the pause in between. The number of words is not specified, it is only known that they all belong to a limited vocabulary set consisting of single word utterances (templates) that were spoken in isolation during the learning phase.

The above problem can be solved by decomposing the matching procedure into two levels: a single template matching level and a word string construction level [5], [7] or by treating the matching procedure as a one-stage procedure [6]. Figure 2 presents a synopsis of this second approach.

Let us consider the plane where an unknown utterance (composed of the words: C, A, D, B, A) is presented along the abscissa, and all the templates along the ordinate. In the second approach the matching procedure is the same as in the case of isolated word recognition, i.e., the goal is to find the best matching patch representing the optimal temporal relations between the parts (words) of the utterance and the templates from the vocabulary set.

The problem presented in Figs. 1 and 2 differ as follows:

the initial point of the path from Fig 2 is unknown; we only know that it coincides with the beginning of any template,

the terminal point of the path from Fig. 2 is also unknown; we only know that it coincides with the end of any template,

the path in Fig. 2 possesses some (generally unknown number) points of discontinuity corresponding to the transitions between the templates,

the minimal accumulated distance $D^*(I-1, J-1)$ is sufficient to recognize the un-

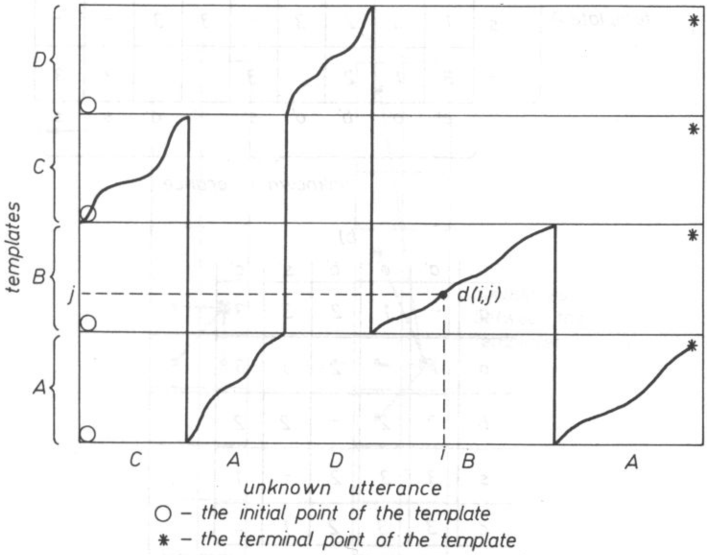
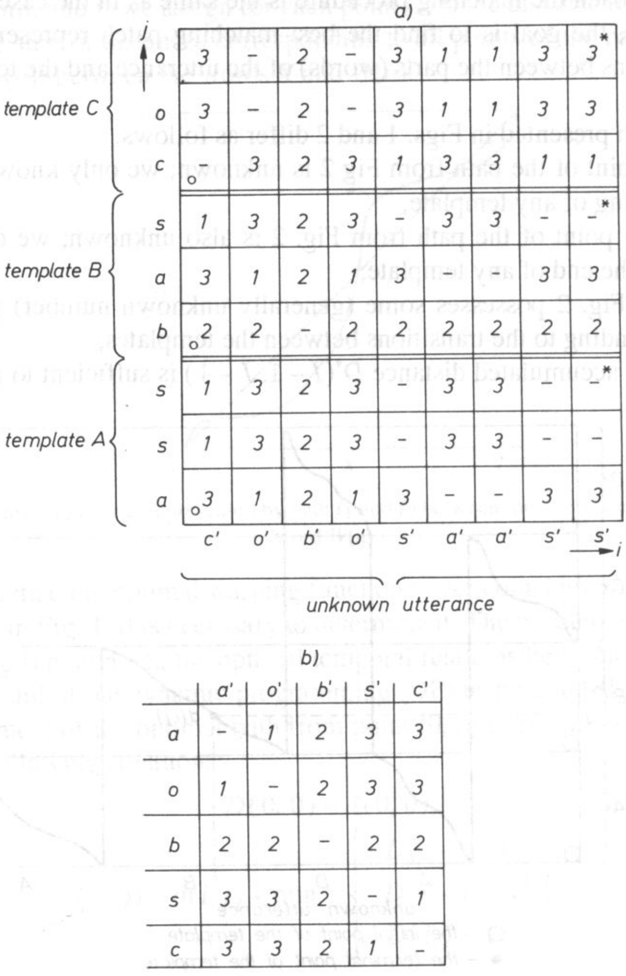

templates
D
C
B
A
j
d(i,j)
i
C
A
D
B
A
unknown utterance
○ – the initial point of the template
* – the terminal point of the template

FIG. 2. Synopsis of the one-stage recognition procedure

known word in Fig. 1, whereas $D^*(I-1, J-1)$ for the optimal path in Fig. 2 gives no information about the unknown sequence of words.

The one common element in both problems (from Figs. 1 and 2) is that the global distance (i.e. the sum of the local distances along the path) is the criterion for the matching procedure. From the optimal path the searched sequence of templates can be uniquely recovered (see Fig. 2).

The method to be described will be illustrated by means of a hypothetical example presented in Fig. 3a. Let us consider the vocabulary set composed of three words: $A = (a\,s\,s)$, $B = (b\,a\,s)$, $C = (c\,o\,o)$, where for the sake of simplicity letter notations are used with reference to the sequence of vectors describing the speech. In the example each of the words consists of three vectors (letters), whereas in reality one vector corresponds to the 10–20 ms. of speech making the words more cumbersome. For the same



a)

	j									
template C	o	3	–	2	–	3	1	1	3	3*
	o	3	–	2	–	3	1	1	3	3
	c	○–	3	2	3	1	3	3	1	1
template B	s	1	3	2	3	–	3	3	–	–*
	a	3	1	2	1	3	–	–	3	3
	b	○2	2	–	2	2	2	2	2	2
template A	s	1	3	2	3	–	3	3	–	–*
	s	1	3	2	3	–	3	3	–	–
	a	○3	1	2	1	3	–	–	3	3
		c'	o'	b'	o'	s'	a'	a'	s'	s' → i

unknown utterance

b)

	a'	o'	b'	s'	c'
a	–	1	2	3	3
o	1	–	2	3	3
b	2	2	–	2	2
s	3	3	2	–	1
c	3	3	2	1	–

FIG. 3. Hypothetical example with an unknown utterance $X = (c'o'b'o's'a'a's's')$ and reference words: A, B, C. Local distances between the vectors are presented in Fig. b.

reason, the unknown utterance X is also presented in the simplified form, i.e. $X = (c' \;\; o' \;\; b' \;\; o' s' a' a' s' s')$.

Following the calculation of all necessary local distances involved according to the data in Fig. 3b where, for example, the distance between the vowels /*a*/ and /*o*/ equals 1, the problem can now be formulated as follows: how to use the dynamic programming technique in order to find such a path connecting one of the initial points (marked by *o*) with one of the terminal points (marked by $*$), for which the sum of local distances along this path attains its minimal value.

In order to use the DP technique, it is necessary to supplement the DP equations (1) for the parts corresponding to the transition from the end of one template to the beginning of the next one. It is evident that the path may attain the point (i, j) (except the initial ones) from the following points: $(i-1, j)$, $(i-1, j-1)$, $(i, j-1)$ (see Fig. 4a). The initial points (see Fig. 2) can be reached from the point $(i-1, j)$, or, which is the above mentioned supplement, from the terminal point of any template including the same template (Fig. 4b). In these cases Eq. (1) must be supplemented with the following equations which applies to the initial points

$$D(i,j) = d(i,j) + \min \begin{cases} D(i-1,j) \\ D(i-1,j_{k,1}) \\ \quad | \\ D(i-1,j_{k,w}) \\ \quad | \\ D(i-1,j_{k,W}) \end{cases} \tag{2}$$

where $j_{k,w}$ indicates the terminal point of the w-th template.

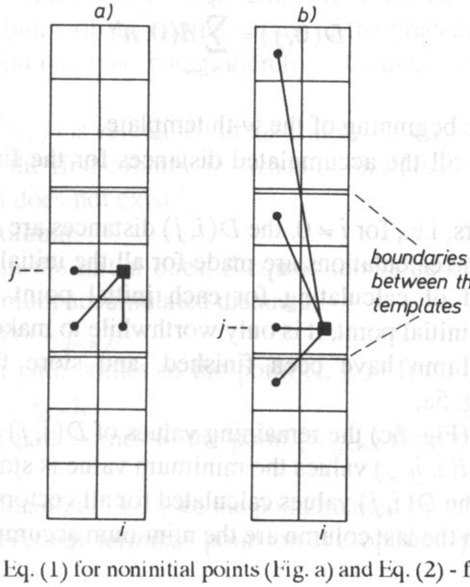


FIG. 4. Illustration of Eq. (1) for noninitial points (Fig. a) and Eq. (2) - for initial ones (Fig. b).

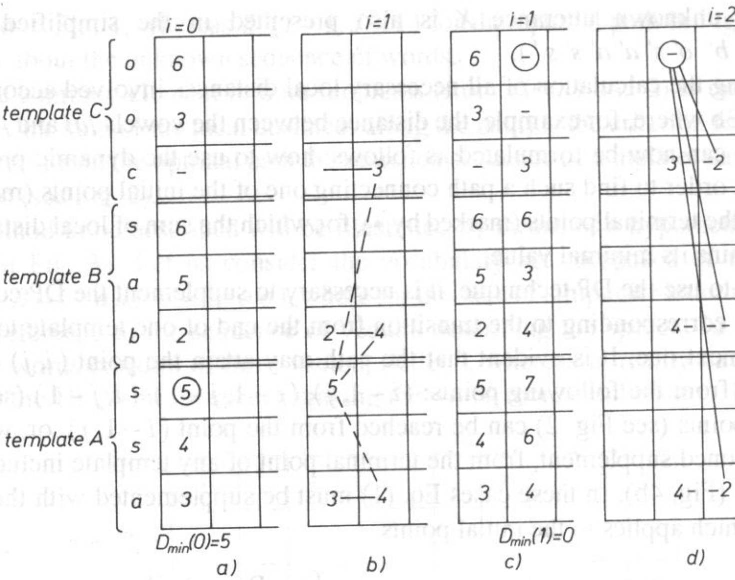


FIG. 5. Some initial steps of the calculations for the example from Fig. 3a.

Figure 5 presents some initial steps of calculations, for the example from Fig. 3a. All the calculations, except the column indexed as $i = 0$, are realized according to Eqs. (1) and (2). For $i = 0$, due to the fact that the optimal path reaches the point $(0, j)$ always from the point $0, j - 1$, except for $j = j_{p,w}$ which is treated as initial for the path, the minimal accumulated distances $D(i, j)$ are calculated as follows:

$$D(0, j) = \sum_{n = j_{p,w}}^{j} d(0, n) \tag{3}$$

where $j_{p,w}$ indicates the beginning of the w-th template.

Figure 5a presents all the accumulated distances for the first vector $(i = 0)$ of the incoming speech.

For all other vectors, i.e., for $i \neq 0$, the $D(i, j)$ distances are calculated in two steps:

at first (Fig. 5b), the calculations are made for all the initial points according to Eq. (2). Note that instead of calculating for each initial point the minimum from all $D(i - 1, j_{k,w})$ for each initial point, it is only worthwhile to make it after the calculations for the preceding column have been finished, and store this minimum value as $D\min(i - 1)$ – see Fig. 5a,

in the second step (Fig. 5c) the remaining values of $D(i, j)$ are calculated according to Eq. (1). Among all $d(i, j_{k,w})$ values the minimum value is stored as $D\min i$.

Figure 6 presents the $D(i, j)$ values calculated for all vectors of unknown utterances. The circled numbers in the last column are the minimum accumulated distances from the

template A	o	6	–	2	2	5	5	5	8	8	
	o	3	–	2	2	5	4	5	8	5	
	c	–	3	2	5	3	4	7	6	2	
template B	s	6	6	4	4	1	4	6	3	3	
	a	5	3	2	1	4	3	3	6	6	
	b	2	4	–	2	4	3	5	7	3	
template A	s	5	7	6	7	3	6	7	1	1	
	s	4	6	4	5	3	4	4	1	1	
	s	3	4	2	3	5	1	1	4	4	
		0	1	2	3	4	5	6	7	8	i
		c'	o'	b'	o'	s'	a'	a'	s'	s'	
		unknown utterance									

FIG. 6. All $D(i, j)$ values for all the vectors of an unknown utterance.

beginning to the end of the utterance, assuming that the last words in the utterance are: A, B or C. It should be noted that $D^*(X, A) = 1, D^*(X, B) = 3, D^*(X, C) = 8$.

Since the minimal accumulated distance applies to the case when the last word is $A\,(D\text{min} = 1)$, it can be recognized as the last word in an unknown sequence. Unfortunately it is impossible to recognize the remaining part of the utterance on the basis of only the data from Fig. 6. To allow for additional data, let us consider the table in Fig. 7a. For each point (i, j), besides $D(i, j)$, it contains the so-called backpointer $i'(i, j)$ which can be defined as the index of the ending vector of the preceding word from which the optimal path to the point (i, j) has come; therefore it indicates the position of the end of the preceding word.

The backpointer $i'(i, j)$ is calculated in the following way:

for all the points of the first column, i.e. when $i = 0$, $i'(0, j) =$ "–". The "–" indicates – "the preceding word does not exist",

for the remaining column:

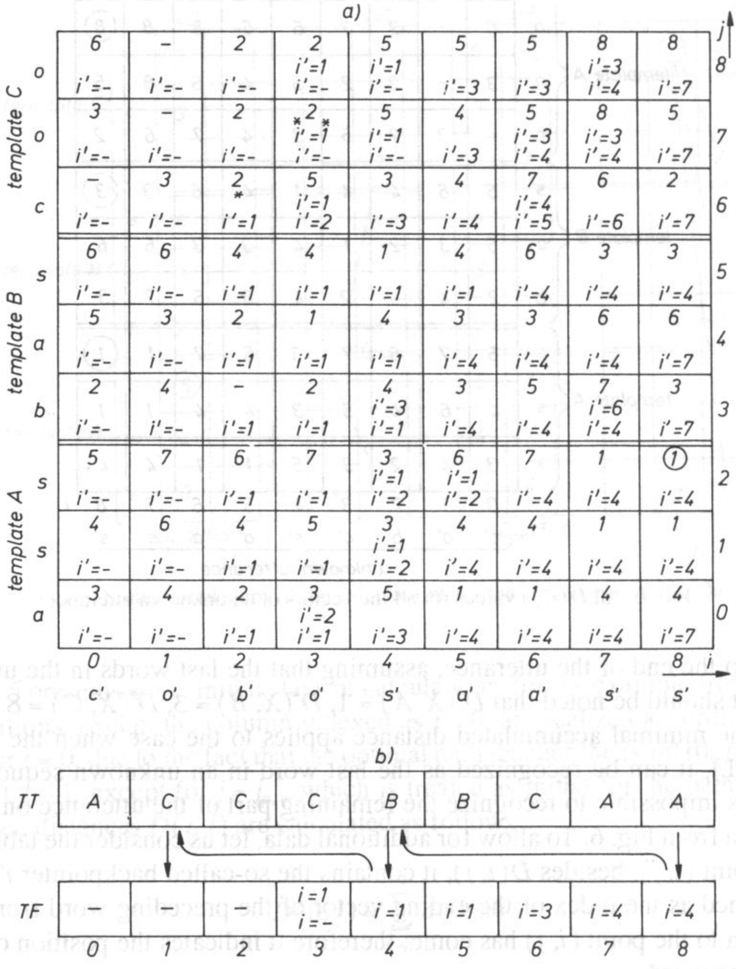
for points other than the initial ones, i' equals the value of i' from the point which Eq. (1) gives the minimum accumulated distance,

for the initial points ($j = j_{p,w}$):

– if the optimal path comes to the point $(i, j_{p,w})$ from the point $(i-1, j_{p,w})$, the $i'(i, j_{p,w})$ equals $i'(i-1, j_{p,w})$,

– if the optimal path comes to the point $(i, j_{p,w})$ from one of the terminal points $(i-1, j_{k,w})$ the $i' = i - 1$.

For example, the notation $i' = 1$ (see the point marked by the asterisk in Fig. 7a) indicates that the last precedent terminal point on the optimal path which comes to the



a)

template C	o	6 i'=-	– i'=-	2 i'=-	2 i'=1 i'=-	5 i'=1 i'=-	5 i'=3	5 i'=3	8 i'=3 i'=4	8 i'=7	8
	o	3 i'=-	– i'=-	2 i'=-	*2* i'=1 i'=-	5 i'=1 i'=-	4 i'=3	5 i'=3 i'=4	8 i'=3 i'=4	5 i'=7	7
	c	– i'=-	3 i'=-	2 * i'=1	5 i'=1 i'=2	3 i'=3	4 i'=4	7 i'=4 i'=5	6 i'=6	2 i'=7	6
template B	s	6 i'=-	6 i'=-	4 i'=1	4 i'=1	1 i'=1	4 i'=1	6 i'=4	3 i'=4	3 i'=4	5
	a	5 i'=-	3 i'=-	2 i'=1	1 i'=1	4 i'=1	3 i'=4	3 i'=4	6 i'=4	6 i'=7	4
	b	2 i'=-	4 i'=-	– i'=1	2 i'=1	4 i'=3 i'=1	3 i'=4	5 i'=4	7 i'=6 i'=4	3 i'=7	3
template A	s	5 i'=-	7 i'=-	6 i'=1	7 i'=1	3 i'=1 i'=2	6 i'=1 i'=2	7 i'=4	1 i'=4	① i'=4	2
	s	4 i'=-	6 i'=-	4 i'=1	5 i'=1	3 i'=1 i'=2	4 i'=4	4 i'=4	1 i'=4	1 i'=4	1
	a	3 i'=-	4 i'=-	2 i'=1	3 i'=2 i'=1	5 i'=3	1 i'=4	1 i'=4	4 i'=4	4 i'=7	0
		0	1	2	3	4	5	6	7	8	i
		c'	o'	b'	o'	s'	a'	a'	s'	s'	

b)

TT	A	C	C	C	B	B	C	A	A
TF	–	–	–	i=1 i=-	i=1	i=1	i=3	i=4	i=4
	0	1	2	3	4	5	6	7	8

FIG. 7. Table of $D(i,j)$ values supplemented by the backpointers $i'(i,j)$ - Fig. a. Tables *TT* and *TF* for the above example - Fig. b.

marked point had appeared in the first ($i = 1$) column. In Fig. 7a the optimal path comes to the point (2, 6) from the terminal point of the template of the word *C* for which the accumulated distance is minimum.

The point marked by a double asterisk presents the situation when two different backpointers must be stored.

Figure 7b presents two additional tables: *TT* and *TF*. Each time after the calculations for the *i*-th vector have been finished, it may be assumed that this is the last vector of the utterance. In the *i*-th position of the table *TT*, i.e., *TT*(i), the number of this template for which the terminal point attains minimum is stored; *TT*(i) indicates the last most likely word in the incoming speech which terminates with the current vector.

In the *TF*(i) the value of the i' from the terminal point which has the minimum $D(i, j_{k,w})$ should be rewritten.

The pair $\{TT(i), TF(i)\}$ indicates which word ends the utterance, assuming that the i-th vector is the last one, and which vector was the last in the precedent word.

After both tables are completed, the recognition process is trivial. In the example from Fig. 7 the position *TF*(8) indicates the word *A* as the last word in the utterance. The *TF*(8) indicates the last vector of the precedent word, which was the 4-th vector. It ended the word *B* because *TT*(4) contains the code of the word *B*. The *TF*(4) indicates the ending vector (*TF*(4) = 1) for the third word from the end of the utterance. It is evident that it was the word *C*, and that no other word preceded the word *C* (*TF*(1) = "–").

Hence the sequence of the templates $\{C\,B\,A\}$ is most likely to be unknown utterance. The accumulated distance *D*min = 1 results from the incorrect analysis of the word "bas" which has been identified as "bos". Note that in Fig. 3 b the $d(a, o) = 1$.

It should be noted that on the basis of the data in the tables *TT* and *TF* more than one decision can be made. Let us suppose that the 4-th vector of the speech agrees with the end of the utterance. From the *TF*(3) it results that the word *C* is the last word in the sequence and that:

– the precedent word does not exist - it applies to the case when *TF*(3) = "–", or

– the precedent word exists and ends with the second vector *TF*(3) = 1. From the *TT*(1) it results that it is also the word *C*.

These decisions can be interpreted as follows: it was the word *C* with the drawled out vowel /o/ (*"cooo"*), or there were two words: *CC ("co co")*. In both cases illustrated in Fig. 8 *D*min = 2 because of $d(b, o) = 2$ (in the first case) or $d(b, c) = 2$ (in the second one).

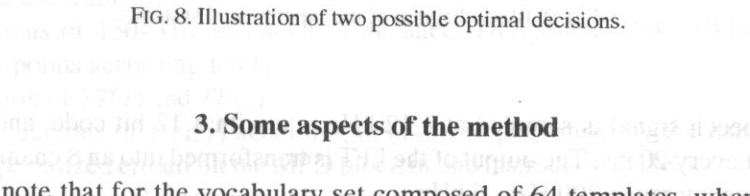


FIG. 8. Illustration of two possible optimal decisions.

3. Some aspects of the method

Let us note that for the vocabulary set composed of 64 templates, when each template is the sequence of 32 vectors, the table in Fig. 7a must have 64 (templates) × 32 (vectors per template) × 250 (number of vectors in the utterance of 5 s. duration) × 2 (parameters: $D(i, j)$, i') = *ca* 1 MB storage locations.

It is easy to mention that the calculations connected with the i-th vector require only the data from the column $i - 1$. It decreases considerably the storage requirements but only on condition that all the calculations are finished before the vector $i + 1$ comes. Most frequently it is maximum 20 ms. For the data as above (64 templates, 32 vectors per template) the time required for $d(i, j)$, $D(i, j)$, $i'(i, j)$ calculations is about 10 μs.

The number of 2048 $d(i, j)$ calculations (64 × 32) can be considerably reduced after the vector quantization (VQ) procedure [2]. The use of rather "economical" 8–bit VQ yielded, for example, in [9] only 0,4% of the recognition quality deterioration. Note that 8-bit VQ implies 8 times fewer $d(i, j)$ calculations - 256 instead of 2048.

The next practical aspect concerns the following theorem: if for the i-th column it occurs that $i' = c$ for all j (see Fig. 7a), then the c-th vector of an utterance is the ending vector of some word in this utterance. The proof is very simple; the identical value of c signifies that independently of the point through which the optimal path crosses the i-th column, the previous word ends with the c-th vector.

The above theorem allows to recognize the word before the end of the whole utterance.

4. Experiments and results

Figure 9 presents a block diagram of the experimental system. This system is based on the IBM PC/AT microcomputer with additional blocks connected to the AT bus:

analog/digital converter ADC,

spectrum analyzer FFT,

specialized block MPD for some DP calculations introduced in order to perform real time recognition for a 10–word vocabulary set.

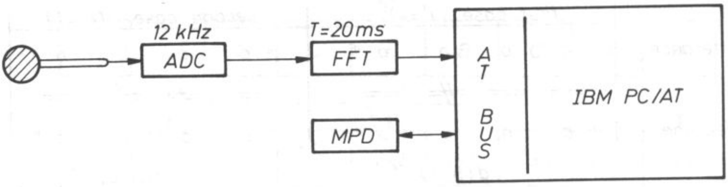


FIG. 9. Block diagram of the experimental system.

The speech signal is sampled at a 12 kHz rate with a 12 bit code, and the FFT is carried out every 20 ms. The output of the FFT is transformed into an 8 channel spectrum in the frequency range 200 Hz ÷ 6 kHz.

In the learning phase (see Fig. 10) the isolated words are stored in memory. After all the words are completed, which implies several hundreds of speech vectors, the 64 cluster search procedure and vector quantization (VQ) procedure are initialized.

In the result, the description of word templates is 8 times reduced; 8 bytes of each spectrum is replaced by a 6-bit number (index) of the nearest cluster obtained by averaging all of the vectors (spectrums) belonging to this cluster.

After the VQ procedure and linear normalization to the length of N numbers ($N = 16 \div 32$), the word templates are stored in MPD block. Another advantage from the

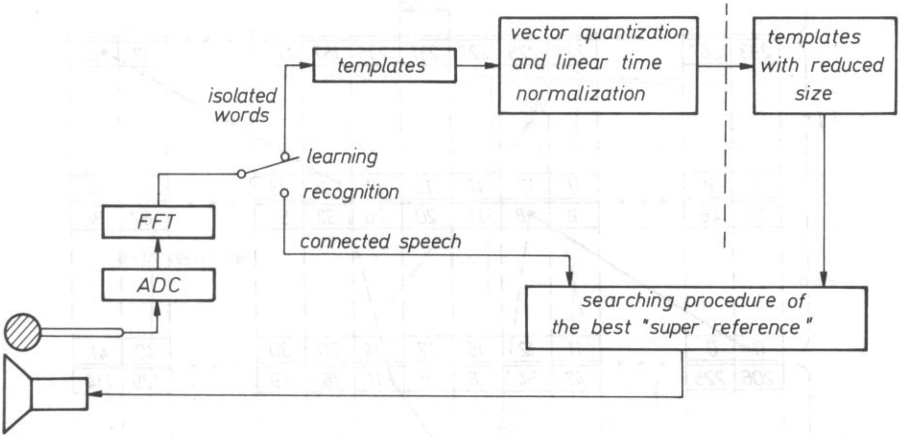

vector quantization and linear time normalization
templates with reduced size
templates
isolated words
learning
recognition
FFT
ADC
connected speech
searching procedure of the best "super reference"

FIG. 10. Functional block-diagram of the experimental system.

VQ procedure is that instead of calculating 160 to 320 local distances between the current vector of incoming speech and all vectors creating word templates (16 to 32 spectrums for each template) only 64 distances should be calculated. These 64 local distances form the table of local distances. The values of $d(i, j)$ in Eq. (1) are taken from the table according to the vector's index in the template.

In our experiment the number of 64 clusters was sufficient from the point of view of the recognition score and was found to be maximum to recognize the connected word in real time.

In the real time systems the recognition procedure starts after the first vector of an utterance is obtained. All the necessary calculations must be finished before the next incoming vector, i.e., in 20 ms. These calculations include:

calculations of 64 local distances,

calculations of 10 accumulated distances $D(i, j)$ and 10 backpointers $i'(i, j)$ for the initial points according to (2),

calculations of 150–310 accumulated distances $D(i, j)$ and 150–310 backpointers for all other points according to (1),

calculation of $TT(i)$ and $TF(i)$.

The calculations of $D(i, j)$ according to (1) were performed by the hardware, realized by a specialized circuit on the MPD block in one instruction cycle of the microcomputer. This was possible because the memory for accumulated distances and backpointers was also situated on the MPD block.

The first experiments demonstrated the need to introduce the template of pause for two reasons:

the recognized utterance can include some short pauses between words,

if a vocabulary set contains words beginning with affricates, the short pause characteristic of such a consonant can appear, which does not exist in other templates. As the example of such a case, Fig. 11 presents the simplified (i.e., containing only the initial and terminal points for each template) table with the $D(i, j)$ values for several vectors of

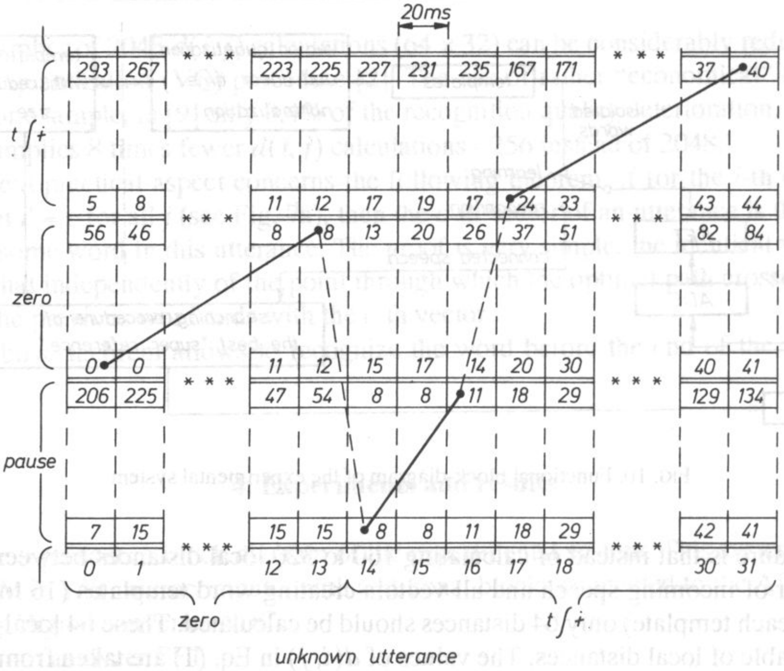


FIG. 11. Part of the table containing $D(i,j)$ values for the utterance /zerot∫ɨ/ and the optimal warping function. The template of the word /t∫ɨ/ uttered in isolation in the learning phase does not contain the pause.

a real utterance "zerot∫ɨi". The optimal path contains the part (vectors 14÷16) corresponding to the 60 ms of the pause made by stopping the air completely at the beginning of the affricate /t∫/. In Fig. 11 only the templates for the pause, and the words "zero", "t∫ɨ" are shown. Since the subject of our experiments was the presented method, and not the parametrization level, only three words were chosen to create the 10 word vocabulary set. It contains the template of pause, and three repetition of these three words: "zero", "t∫ɨ", "t∫terɨ".

The recognized set contained all (81) combinations in four-word utterances repeated twice by the same speaker.

The results of the recognition of 162 sentences are presented in Table 1.

In the first experiment the normalized length of each template was equal to 32 vectors, and the "block city" metric was used to calculate the distance between either two vectors. In this experiment 10 words were deleted.

The detailed analysis has shown that these deletions occur when the word uttered in a phrase is much shorter than its templates. Figure 12 presents an example when a part of the phrase contains two of the same words, but the first one was uttered in avery fast manner. A much smaller number of local distances on the path d_1 towards the path d_N, can result in $D1 < DN$, and in effect the optimal path runs along the path marked by the solid line rather than along the dashed line. The first word will be deleted because the backpointer in the point $(1, i+1)$ will be different from i.

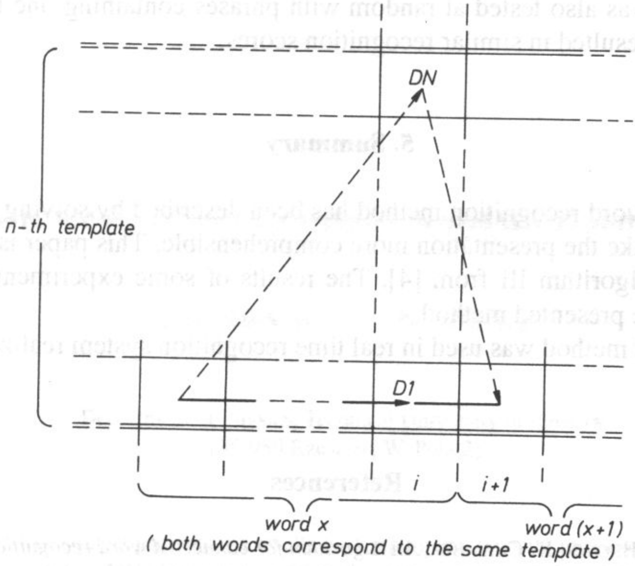


FIG. 12. Simplified figure for a word deletion mechanism.

The above case was caused by the parametrization level (8 digital filters), for which the average distances between the vectors belonging to different phonetic classes were insufficiently greater than the distances between the vectors belonging to the same phonetic class.

In order to confirm this fact in the second experiment the nonlinear "city" metric was used. The distances smaller than a certain threshold was replaced by zero. The threshold was chosen as the average distance between the vectors belonging to the same phonetic class. The above nonlinearity diminished the number of deletions to 4 cases.

Another way to solve the problem connected with quickly uttered words consists in experiments III and IV in shortening the length of the templates to 16 vectors instead of 32. In experiment IV with nonlinear "city" metric any deletion was noted. The recognition score for the phrases was about 99,4% - the word "zero" was misidentified as "tʃterɨ".

Table 1. Results of the recognition of 162 four-word sentences

Experiment	Normalized length	Metric	Deleted	Misidentified
I	32	lin.	10	–
II	32	nonlin.	4	–
III	16	lin.	3	–
IV	16	nonlin.	–	1

The method was also tested at random with phrases containing one to nine words. These tests have resulted in similar recognition scores.

5. Summary

A connected word recognition method has been described by solving the hypothetical problem to make the presentation more comprehensible. This paper is a detailed description of the algorithm III from [4]. The results of some experiments confirm the effectivness of the presented method.

The described method was used in real time recognition system realized under grant CPBR 7.1.

References

[1] J. BRIDLE, M. BROWN, R. CHAMBER, *An Algorithm for connected word recognition*, Proc. ICASSP 1982, pp. 899–902.

[2] R.M. GRAY, *Vector quantization*, IEEE ASSP Mag., **1**, 2, pp. 4–29 (1984).

[3] S. GROCHOLEWSKI, *Dynamic programming in automatic speech recognition* (in Polish), Elektrotechnika (AGH), **4**, 1, pp. 106–126 (1985).

[4] S. GROCHOLEWSKI, *Algorithms for connected word recognition – A global approach*, Archives of Acoustics, **16**, 3/4, pp. (1991).

[5] C.S. MYERS, L.R. RABINER, *A level building dynamic time warping algorithm for connected word recognition*, IEEE Trans. ASSP, **29**, 2, pp. 284–297 (1987).

[6] H. NEY, *The use of a one-stage dynamic programming algorithm for connected word recognition*, IEEE Trans. ASSP, **32**, 2, pp. 263–271 (1984).

[7] H. SAKOE, *Two-Level DP Matching – A dynamic programming based pattern matching algorithm for connected word recognition*, IEEE Trans. ASSP, **27**, 6, pp. 588–595 (1979).

[8] H. SAKOE, S. CHIBA, *Dynamic programming optimization for spoken word recognition*, IEEE Trans. ASSP, **26**, 2, pp. 43–49 (1978).

[9] N. SUGAMURA, S. FURUI, *Speaker-Dependent Large Vocabulary Word Recognition Using SPLIT Method*, Review of the Electrical Communications Laboratories, **34**, 3, pp. 327–332 (1986).

[10] T.K. VINTSYUK, Element-wise recognition of continuous speech composed of words from a specified dictionary, Kibernetika, **7**, 2, pp. 133–143 (1971).

Received on January 10, 1990