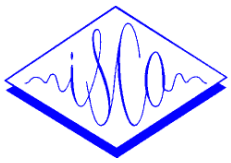


First International Conference on
Spoken Language Processing (ICSLP 90)
Kobe, Japan
November 18-22, 1990

SPEAKER-INDEPENDENT WORD SPOTTING AND A TRANSPUTER-BASED IMPLEMENTATION

Akihiro IMAMURA and Yoshitake SUZUKI

NTT Human Interface Laboratories
Yokosuka-shi, Kanagawa, 238-03 JAPAN

ABSTRACT

This paper describes an HMM-based speaker-independent word spotting system and its Transputer-based implementation. The candidates of word end-points and the corresponding likelihood scores are computed with the continuous Viterbi decoding algorithm. To prune unreasonable candidates, a new duration control method, a threshold logic for the likelihood scores and a new local peak detection method are proposed. An efficient parallel processing scheme for the word spotting system is carried out by using a tree structure of Transputers. In each frame period, the spectral feature vector from the speech analyzer is broadcasted from the root Transputer (Processing Master : PM) to the node Transputers (Processing Element : PE). Each PE performs the continuous Viterbi decoding and the pruning of candidates in parallel, and the spotting results are returned to PM. With 8 PEs in a tree structure, 72 words can be processed within a 12msec frame period. Word detection experiments, using the 10 Japanese digits spoken over a noisy telephone network, yield a word detection accuracy of 97%.

1. INTRODUCTION

HMMs (Hidden Markov Models) have been successfully applied to various constrained tasks, such as isolated word recognition and continuous speech recognition [1][2]. In particular, several kinds of HMM-based isolated word recognizers have been put into practical use.

When the inputs to such isolated word recognition systems include noises and/or non-vocabulary words, satisfactory recognition performance can not be achieved. The solution of this problem requires a word spotting system which uses a vocabulary to locate words embedded in the input speech [3][4].

Our speaker-independent word spotting system is based on HMM word modeling with fuzzy vector quantized observation symbols and pruning of unreasonable candidates. The continuous Viterbi decoding algorithm is employed to obtain the candidates of word end-points and the corresponding likelihood scores. Unreasonable candidates are pruned by two validity tests performed in a postprocessor. The tests utilize a new duration control method and a threshold logic for the likelihood scores. From the set of reasonable candidates, only the candidates which have locally peaked likelihood scores are chosen as the spotting results. An efficient parallel processing scheme for the proposed word spotting algorithm is also carried out by using a tree structure of Transputers.

In the following sections, the details of our word spotting system and the Transputer-based prototype are presented, followed by experimental results.

2. HMM-BASED WORD SPOTTING SYSTEM

The block diagram of the HMM-based word spotting system is shown in Fig.1. There are essentially three steps in the system: acoustical preprocessing, including LPC analysis and fuzzy vector quantization (fuzzy VQ); continuous Viterbi decoding; and postprocessing.

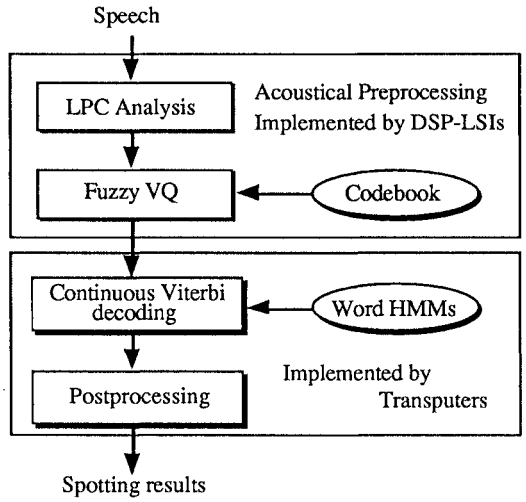


Fig. 1 Block diagram of the word spotting system.

2.1 Acoustical preprocessing

In the acoustical preprocessor, the input speech signal is converted to a set of log-scale delta-energy, LPC derived cepstral and delta-cepstral vectors. Each spectral feature vector is then coded to fuzzy observation symbols using fuzzy VQ [5]. Fuzzy VQ represents an input vector as a weighted combination of the code-vectors. A set of the weighting functions, called the membership function, becomes the fuzzy observation symbols. The membership function is given as follows:

$$u_{t,k} = \left\{ \sum_{j=1}^{M} \left[\frac{d_{t,k}}{d_{t,j}} \right]^{\frac{1}{F-1}} \right\}^{-1}, \qquad (1)$$

where
$u_{t,k}$: membership function,
$d_{t,k}$: distance between input vector at frame t and code-word k,
M : codebook size,
F : fuzziness (< 1).

By representing input vectors as fuzzy observation symbols, we can obtain accurate word HMMs and high recognition accuracy with a small number of training tokens [6].

2.2 Word HMMs

The parameters of word HMMs are estimated by using the maximum likelihood *forward-backward* procedure. For each word model, we used the simple *"left-to-right"* type topology without arcs which skip states. An initial state distribution π_i is assigned to each state. An arc from state i to state j is assigned a log-scale state transition probability $a_{i,j}$ and a set of discrete observation symbol probabilities $\{b_{i,j}(k)\}$. To obtain accurate observation symbol probabilities for each input spectral feature vector x_t, log-scale interpolated observation symbol probabilities $\omega_{i,j}(x_t)$ are computed using membership functions and discrete observation symbol probabilities as follows:

$$\omega_{i,j}(x_t) = log \left\{ \sum_{k=1}^{M} u_{t,k}\, b_{i,j}(k) \right\} \quad (2)$$

2.3 Continuous Viterbi decoding

Against whole vocabulary words, input speech is scored frame-synchronously by using the continuous Viterbi decoding algorithm. In each frame period, the selection of an optimal state transition is performed by comparing the likelihood scores normalized by the length of the corresponding state transition path. The details of the algorithm are given as follows:

Definition of variables and constants.

$\delta_t(j)$: The likelihood of optimal path to state j at frame t.
$B_t(j)$: The beginning-point of the optimal path.
$\Psi_t(j)$: The index of optimal state at frame t-1.
N : Number of state.
T : The index of current frame.

step 1 - Initialization.

For $1 \le i \le N$

$$\delta_0(i) = \pi_i \quad (3)$$

$$B_0(i) = 1 \ : \ if \ \delta_0(i) \neq -\infty \quad (4)$$

step 2 - Recursion.

For $1 \le t \le T$, $1 \le j \le N$

$$i' = \underset{1 \le i \le N}{argmax} \left\{ \frac{\delta_{t-1}(i) + a_{i,j} + \omega_{i,j}(x_t)}{t - B_{t-1}(i) + 1} \right\} \quad (5)$$

$$\delta_t(j) = max \left\{ \delta_{t-1}(i') + a_{i',j} + \omega_{i',j}(x_t) \ , \ \pi_j \right\} \quad (6)$$

$$B_t(j) = \begin{cases} B_{t-1}(i') & : \ if \ \delta_t(j) \neq \pi_j \quad (7) \\ t+1 & : \ if \ \delta_t(j) = \pi_j \quad (8) \end{cases}$$

$$\Psi_t(j) = i' \ : \ if \ \delta_t(j) \neq \pi_j \quad (9)$$

step 3 - Termination at t = T.

$i_T = \underset{1 \le i \le N}{argmax} \{ \delta_T(i) \}$: The index of optimal final state. (10)

$B_T(i_T)$: The index of beginning-point for end-point T. (11)

$$P(T) = \frac{\delta_T(i_T)}{T - B_T(i_T) + 1} \ : \text{The likelihood score.} \quad (12)$$

step 4 - Back-tracking of optimal state transition history.

For $t = T\text{-}1, T\text{-}2, \ldots, B_T(i_T)\text{-}1$

$i_t = \Psi_{t+1}(i_{t+1})$: The index of optimal state at frame t. (13)

The results of this decoding are the candidates of word beginning-point $B_w(t)$, likelihood $P_w(t)$ and state transition history $I_w(t)$ for each word w and each word end-point t.

2.4 Postprocessing

2.4.1 Pruning by duration control

HMMs are effective stochastic models to express speech signals, but phoneme duration information is not modeled suitably in the ordinary HMMs. Because of this, candidates which have unreasonable state transition histories sometimes appear in recognition results. To overcome this problem, HMM state and word duration control methods based on the statistical information of training tokens were introduced as follows:

step 1 - Estimation of HMM state and word duration range.

Generally, phoneme duration is a function of word duration. Therefore, the 2nd-order regressive function of word duration Tw is assumed to be an HMM state duration distribution $d(w,i)$ as follows:

$$d(w,i) = b0(w,i) + b1(w,i)\, Tw + b2(w,i)\, Tw^2. \quad (14)$$

In each state i of word w, the regression coefficients $b0(w,i)$, $b1(w,i)$, $b2(w,i)$ and the standard deviation of state duration $d_{dev}(w,i)$ are estimated using the optimal state transition history of training tokens. The statistical constants of word duration (minimum : Tw_{min}, maximum : Tw_{max} and standard deviation : Tw_{dev}) are also estimated from training tokens.

step 2 - Pruning by HMM state duration range.

During the recognition process, the state duration $D(w,i)$ of each candidate word w is computed by back-tracking the corresponding optimal state transition history. If one or more $D(w,i)$ do not satisfy the following equation,

$$d(w,i) - r_1\, d_{dev}(w,i) < D(w,i) < d(w,i) + r_1\, d_{dev}(w,i), \quad (15)$$

where r_1 is a constant, the candidate is pruned.

step 3 - Pruning by word duration range.

Pruning based on the word duration is also performed as follows. If the word duration Tw does not satisfy the following equation,

$$Tw_{min} - r_2\, Tw_{dev} < Tw < Tw_{max} + r_2\, Tw_{dev}, \quad (16)$$

where r_2 is a constant, the candidate is pruned.

2.4.2 Pruning by likelihood score

Candidate pruning by threshold logic for the minimum of likelihood scores is effective to reduce the number of unreasonable candidates at noise and/or non-vocabulary word periods. As in the duration control, the statistical constants of likelihood scores (minimum : Pw_{min} and standard deviation : Pw_{dev}) are estimated using the likelihood scores from training tokens. Only the candidate which has a greater likelihood score than the minimum threshold score Pw_{th} given by

$$Pw_{th} = Pw_{min} - r_3\, Pw_{dev}, \quad (17)$$

where r_3 is a constant, is accepted as a reasonable candidate.

2.4.3 Local peak detection

Sometimes, there are several candidates which indicate the same word closely located in the set of reasonable candidates output by the previous pruning methods. To reduce such redundant candidates, we select the candidate which has the locally peaked likelihood score. The local peak point can be detected by a partial sorting of identical candidate words around the turning point of the sign of the derivative of the regression line for likelihood scores. The sign of the derivative of the

regression line for likelihood scores at frame t is given by

$$S_w(t) = sign\left\{\sum_{j=-L}^{L} j\,P_w(t+j)\right\}, \quad (18)$$

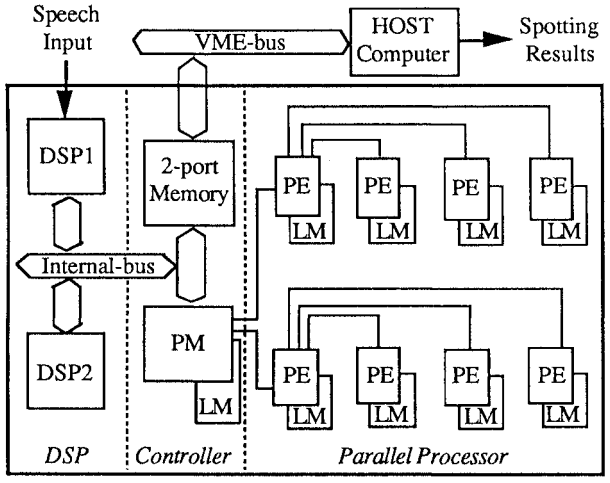
where L is constant.

3. TRANSPUTER-BASED IMPLEMENTATION OF SYSTEM

The word spotting system is prototyped with DSP-LSIs for LPC analysis and fuzzy VQ, and a tree structure of Transputers [7] for continuous Viterbi decoding and postprocessing. In this section, the hardware architecture and the real-time parallel processing scheme are described.

3.1 Hardware architecture

The hardware architecture of the system is shown in Fig.2. The system is divided into three parts. In the DSP part, two commercially available DSP-LSIs are used to carry out the LPC analysis and the fuzzy VQ processing in real-time. The controller mainly consists of a Transputer acting as the processing master (PM), local memory (LM) of PM, and dual-port memory. Other than system control, the PM manages the data transfer between the DSPs and the parallel processor by using the internal-bus and the serial links. Communications between PM and the host computer can be done via the dual-port memory and VME-bus interface. Some of the postprocessing operations are also done by the PM. The parallel processor consists of eight processing elements (PEs) and their local memories. A single Transputer is used for each PE. Efficient connection of all PEs can be carried out by using the serial communication links of each Transputer.



PM : Processing Master, PE : Processing Element,
LM : Local Memory

Fig.2 Hardware architecture of the system.

3.2 Parallel processing

The following procedure must be performed within one frame period for real-time word spotting.

(a)PM receives fuzzy VQ output from the DSP and broadcasts data to each PE in the parallel processor.

(b)Each PE performs word spotting and sends the results back to PM in parallel.

(c)PM transmits the results to the host computer through the dual port memory.

The tree structure of PEs as shown in Fig.2 has been adopted to reduce the inter-PE data communication time. Thanks to the parallel communication function of each Transputer, the data communication time can be efficiently reduced. To reduce the data communication count during the word spotting procedure, each PE holds the complete parameters of one vocabulary word HMM set.

3.3 Processing element (PE)

The two upper PEs receive fuzzy VQ data from the PM and broadcast them to the lower PEs. Each PE has the entire parameters for each word, and the memory requirement per word is about 33Kbytes. Equations (2) through (13) for continuous Viterbi decoding and equations (14) through (17) for pruning are performed in each PE against the vocabulary words in parallel. After processing terminates, the lower PEs send the results to the upper PEs. Each upper PE sends the full results to PM, some of which were obtained by itself and the others by the lower PEs.

3.4 Processing master (PM)

PM performs local peak detection as the final pruning step and vocabulary selection for spotting in addition to data communication and system control.

The processing time for pruning is about 30% of that for continuous Viterbi decoding. The number of words processed in each PE could be increased if all pruning steps were performed in the PM. However, PM performance is limited and the maximum number of candidates which can be processed by a single PM would be only 60. Therefore, local peak detection is performed in the PM for well balanced process partitioning in this system.

When the expected utterance is limited to a subset of the entire vocabulary, higher spotting accuracy could be obtained by selecting the correct subset vocabulary. In this system, the host sends the information for vocabulary selection to the PM before commencing spotting operations. PM broadcasts the information to each PE in order to reduce the number of spotting words in each PE and so obtain higher spotting accuracy.

4. EXPERIMENTAL EVALUATION

In this section, the experimental results of the prototype system are presented. The experiments were aimed at examining the effect of postprocessing (candidate pruning methods) on performance and measuring the real time processing ability of the hardware architecture.

4.1 Speech databases

All the experiments were performed in a speaker-independent mode using the ten Japanese digits; (*/ichi/, /ni/, /san/, /yon/, /go/, /roku/, /nana/, /hachi/, /kyuu/, and /zero/*) as a keyword vocabulary. To training the system, a database containing 107 isolated tokens per word from 107 male speakers were used. All the training tokens were sampled by using the automatic speech end-point detection algorithm. The obtained end-points were then checked and modified manually. For system testing, a database which consisted of short sentences containing 100 tokens from 10 new male speakers was used. Each testing token included one keyword from the vocabulary. Noise and/or some redundant words were also included prior to and/or following the keyword. All speakers uttered tokens passed through several long-distance telephone links which had an *SNR* level of 10dB to 20dB. Both databases were digitized at 8kHz and windowed using a 192-point Hamming window every 12msec. Then, 12-order LPC analysis and fuzzy vector

quantization (*fuzziness = 1.5*) using 256-size codebook was carried out in the acoustical preprocessing phase.

4.2 Recognition performance

Using the training tokens, an 11-state HMM was estimated for each keyword in the vocabulary. The word spotting results on the testing tokens of short sentences are given in Table 1 and Table 2. Table 1 shows word detection accuracies R_c defined by

$$R_c = \frac{(Total\ of\ correct\ detection)}{(Total\ of\ input)}. \quad (19)$$

Table 2 shows false-alarm rates R_a defined by

$$R_a = \frac{(Total\ of\ insertion\ and\ substitution)}{(Total\ of\ detection)}. \quad (20)$$

In both tables, recognition performance was presented using several combinations of values for the pruning control constants (r_1, r_2, r_3) in equations (15), (16) and (17). In table 2, false-alarm rates were computed at the corresponding ranks of candidate in table 1. The local peak detection, constant L in equation (18), was set at 5.

Table 1 Word detection accuracies at several pruning conditions : Rc (%)

Condition of pruning			Rank of candidate		
r1	r2	r3	1	2	3
1.0	1.0	1.0	94.0	96.0	96.0
3.0	3.0	3.0	97.0	99.0	99.0
5.0	5.0	5.0	96.0	99.0	99.0
non	1.0	1.0	96.0	99.0	99.0
non	3.0	3.0	96.0	99.0	99.0
non	5.0	5.0	96.0	99.0	99.0
1.0	non	1.0	95.0	96.0	96.0
3.0	non	3.0	97.0	99.0	99.0
5.0	non	5.0	96.0	99.0	99.0
1.0	1.0	non	94.0	97.0	97.0
3.0	3.0	non	97.0	99.0	99.0
5.0	5.0	non	96.0	99.0	99.0
non	non	non	96.0	99.0	99.0

Table 2 False-alarm rates at several pruning conditions : Ra (%)

Condition of pruning			Rank of candidate		
r1	r2	r3	1	2	3
1.0	1.0	1.0	2.2	1.1	1.1
3.0	3.0	3.0	3.0	1.0	1.0
5.0	5.0	5.0	4.0	1.0	1.0
non	1.0	1.0	4.0	1.0	1.0
non	3.0	3.0	4.0	1.0	1.0
non	5.0	5.0	4.0	1.0	1.0
1.0	non	1.0	2.2	1.1	1.1
3.0	non	3.0	3.0	1.0	1.0
5.0	non	5.0	4.0	1.0	1.0
1.0	1.0	non	6.0	3.0	3.0
3.0	3.0	non	3.0	1.0	1.0
5.0	5.0	non	4.0	1.0	1.0
non	non	non	4.0	1.0	1.0

When each pruning control constant was fixed at 3.0, the best word detection accuracy of 97% for the top candidate and 99% for the top two and three candidates were realized. Without candidate pruning (only local peak detection), the word detection accuracy was 96% for the top candidate, and the false-alarm rate increased 1% from the previous best condition. These results indicate that the proposed pruning methods are effective for accurate word spotting. However, when the value of 1.0 is used for both pruning control constants, the word detection accuracy goes down 3%. To reduce this kind of over-pruning, the pruning control constants should be optimized experimentally.

4.3 Real time processing ability

The processing time of PEs and PM were measured experimentally to clarify the real time processing ability. The processing time per word for continuous Viterbi decoding and pruning was about 936 μsec and 273 μsec respectively using a 20 MHz Transputer. The parallel transmission time of fuzzy VQ results from the upper PE to the lower PEs is about 22 μsec, whereas the parallel transmission time for spotting result from the lower PEs to the upper PE is less than 15 μsec per word. This means that 72 words are processed by 8 PEs within a 12msec frame period. Though the capability of the parallel processor would be improved if more PEs were added to the tree structure, the processing capability of each PE would be reduced because of the increase in inter-PE data communication time.

5. SUMMARY

We have described an HMM-based word spotting system and its Transputer-based hardware implementation. To obtain candidates, the continuous Viterbi decoding algorithm is performed on fuzzy vector quantized observation symbols. For accurate word spotting, two candidate pruning methods, duration control and threshold logic for likelihood scores were introduced.

To construct efficient system hardware, a parallel processing scheme was proposed that uses a tree structure of Transputers. The root Transputer (Processing Master : PM) manages data transfer and performs local peak detection. Each node Transputer (Processing Element : PE) performs continuous Viterbi decoding and candidate pruning in parallel.

Word detection experiments were carried out using short sentences containing the ten Japanese digits as keywords. The proposed pruning methods yielded the best word detection accuracy of 97% at an SNR level of 10dB to 20dB.

The real time processing ability of the proposed parallel processing scheme for the system was also evaluated by experiments. In the case of a tree structure with 8 PEs, 72 words could be processed within a 12msec frame period.

Acknowledgment

The authors wish to thank *Mr. Nobuo Naganuma* for his Transputer programing.

References

[1] Averbuch, A., et. al., "Experiments with the Tangora 20,000 Word Speech Recognizer", ICASSP-87, April, 1987, pp. 701-704.

[2] Lee, K. F., et. al., "Large-Vocabulary Speaker-Independent Continuous Speech Recognition using HMM", ICASSP-88, April, 1988, pp. 123-126.

[3] Kimura, T , et. al., "A Telephone Speech Recognition System Using Word Spotting Technique Based on Statistical Measure", ICASSP-87, April, 1987, pp. 1175-1178.

[4] Rohlicek, J. R., et. al., "Continuous Hidden Markov Modeling for Speaker-Independent Word Spotting", ICASSP-89, May, 1989, pp. 627-630.

[5] Tseng, H, et. al., "Fuzzy Vector Quantization applied to Hidden Markov Modeling", ICASSP-87, April, 1987, pp. 641-644.

[6] Imamura, A., et. al., "Speaker-Independent Word Recognition Through Telephone Networks Using Hidden Markov Models", Eurospeech-89, Sept., 1989, pp. 171-174.

[7] "The Transputer Databook", INMOS Databook Series, Nov.,1988.