A Parallel Phoneme Recognition Algorithm Based on Continuous Hidden Markov Model*

Sang-Hwa Chung, Min-Uk Park, +Hyung-Soon Kim
Dept. of Computer Engineering, Pusan National University
+Dept. of Electronics Engineering, Pusan National University
Pusan, Korea
{shchung,ukui,kimhs}@hyowon.cc.pusan.ac.kr

Abstract

This paper presents a parallel phoneme recognition algorithm based on the continuous Hidden Markov Model (HMM). The parallel phoneme recognition algorithm distributes 3-state HMMs of context dependent phonemes to the multiprocessors, computes output probabilities in parallel, and enhances the Viterbi beam search with a message passing mechanism. The algorithm is implemented in a multi-transputer system using distributed-memory MIMD multiprocessors.

Experimental results show the feasibility of the parallel phoneme recognition algorithm in constructing a real-time parallel speech recognition system based on time-consuming continuous HMM.

1. Introduction

Although the demand for continuous speech recognition systems [10] is increasing, the solution to precisely identify human speech has not yet been discovered. Up to now, the most popular approach in speech recognition is the Hidden Markov Model (HMM) [6,9,11,12]. In particular, continuous HMM based on continuous probability distributions has better recognition rates than discrete HMM using vector quantization preprocessing. However, time-consuming output probability computation of continuous HMM [4] makes it difficult to implement a real-time speech recognition system. In general, current speech recognition systems adopted a sub-optimal beam to the Viterbi search [7] in order to reduce the computational overload in continuous HMM. The more powerful approach is adopting parallel techniques and enlarging the beam width for better recognition accuracy.

* This work was supported by the Ministry of Information and Communication of Korea.

Not many researches are proceeded in the parallel speech recognition area. Some of the relevant researches are as follows. Huijsen [3] performed experiments in modeling and recognizing discrete HMM for speaker dependent model on the nCube2 machine. AT&T [13] proposed a parallel speech recognizer using a multi-threading method under a Silicon Graphics Power Challenge XL shared-memory multiprocessors. Chung and Moldovan [1] implemented a parallel parser for processing spoken language on the SNAP MPP system built for AI applications. Giachin and Rullent [2] devised a parallel parser for spoken language on a transputer-based distributed architecture.

This paper presents a parallel phoneme recognition algorithm based on continuous HMM, which can be a basis for a real-time parallel speech recognition system. Distributing HMMs to parallel processors helps time-consuming output probability computations to be performed effectively. The Viterbi search is enhanced by using a message passing mechanism based on a multi-transputer system. The workload unbalance introduced by the Viterbi beam search is resolved by applying a load balancing scheme. Experimental results show that the parallel phoneme recognition algorithm is effective in constructing a real-time parallel speech recognition system.

The rest of this paper is organized as follows. Section 2 describes continuous HMM and phoneme modeling for experiments. Section 3 describes the Viterbi search and the Viterbi beam search for phoneme recognition. Section 4 shows the parallel phoneme recognition algorithm and the parallel machine in which the algorithm is implemented. Experimental results and conclusions are reported in Section 5 and Section 6 respectively.

2. Phoneme Models Based on Continuous Hidden Markov Model

Hidden Markov Model is an extended stochastic model

from Markov chain to find a sequence of states with observation value, a probabilistic function of a state. (1) shows parameters of a HMM

$\lambda = \{ A, B, \pi \}$ (1)

$A = \{ a_{ij} \}$ The set of transition probability from state i to state j

$B = \{ b_j(k) \}$ The set of observation probability given the current state j

$\pi = \pi_i$ The set of initial state probability

In continuous HMM, $b_j(\boldsymbol{x})$ in (2) is substituted for $b_j(k)$ in (1),

$$b_j(x) = \sum_{k=1}^{M} c_{jk} b_{jk}(x) = \sum_{k=1}^{M} c_{jk} N(x, \mu_{jk}, \sum_{jk}) \qquad (2)$$

Where $\boldsymbol{x}$ is the observation vector being modeled, c_{jk} is the mixture coefficient for the k-th mixture in state j, and N is the Gaussian pdf with mean vector $\boldsymbol{\mu}_{jk}$ and covariance matrix $\boldsymbol{\Sigma}_{jk}$ for the k-th mixture component in state j. The mixture gains c_{jk} satisfy the stochastic constraint :

$$\sum_{k=1}^{M} c_{jk} = 1, \qquad 1 \le j \le \text{number of states}$$

$$c_{jk} \ge 0, \qquad 1 \le j \le \text{number of states},$$
$$1 \le k \le \text{number of mixtures}.$$

As known in (2), the calculation time of output probability $b_j(\boldsymbol{x})$ increases in proportion to the number of mixtures and states. In fact, the calculation of output probability $b_j(\boldsymbol{x})$ is the most time-consuming work in phoneme recognition based on continuous HMM for the reason that the $b_j(\boldsymbol{x})$ must be calculated at every state of each frame.

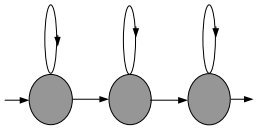

Figure 1 The HMM structure representing a phoneme

Performance and computational complexity of phoneme recognition vary depending on the components of HMM such as number of states, mixtures, and topology of HMM. Furthermore, it requires time-consuming efforts to find the best HMM parameters for recognizing phonemes. Figure 1 shows the structure of a 3-state left-right HMM used in our experiments. For our experiments HMM parameters were modeled from 48 Korean context independent phonemes (CI phoneme), which is retrained as context dependent phonemes (CD phoneme). CD phoneme represents every phoneme that can be positioned between specific phonemes.

As a task domain, the Computer Secretary agent domain data among KOREAN SPEECH DB (ETRI Wonkwang SPEECH DB) [8] is selected for the training and recognition experiments. KOREAN SPEECH DB contains the training data of 100 persons and the test data of 10 persons with 16KHz bandwidth and 16bits quantization. The MFCC (Mel-Frequency Cepstral Coefficients) analysis is adopted to obtain speech features at each frame.

3. The Parallel Viterbi Beam Search

The Viterbi search finds the best phoneme sequence among trained HMMs for the input speech feature vectors. To find the single best state sequence, $q=(q_1 q_2 \ldots q_t)$, for the given observation sequence $O=(o_1 o_2 \ldots o_T)$, we need to define the quantity $\delta_t(i)$ in (3)

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1 q_2 \ldots q_{t-1}, q_t = i, o_1 o_1 \ldots o_t \mid \lambda] \quad (3)$$

$\delta_t(i)$ is the highest score along a single path, at time t, which accounts for the first t observations and ends in state i. By induction we have

$$\delta_{t+1}(j) = \max_i [\delta_t(i) a_{ij}] b_j(o_{t+1}) \quad (4)$$

$b_j(o_{t+1})$ means (2) in Section 2. To retrieve the state sequence, we need to keep track of the argument that maximizes $\delta_{t+1}(j)$ in (4) for each t and j. To store the argument, an array, $\psi_t(j)$ is needed in the algorithm. The Viterbi search is as follows.

1. Initialization

$\delta_1(i) = \pi_i b_j(o_1)$ $\quad 1 \le i \le N$ (states)

$\psi_1(i) = 0$

2. Recursion

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] b_j(o_t) \quad 2 \le t \le T, 1 \le j \le N \quad (5)$$

$$\psi_t(j) = \arg \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}] \quad 2 \le t \le T, 1 \le j \le N$$

3. Termination

$$P^* = \max_{1 \le i \le N} [\delta_T(i)] \qquad q_T^* = \arg \max_{1 \le i \le N} [\delta_T(i)]$$

4. Path (state sequence) backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T-1, T-2, \ldots, 1$$

In continuous HMM, $b_j(o_t)$ in (5) is substituted with output probability $b_j(\boldsymbol{x})$ in (2). The described Viterbi search can be implemented by taking logarithms of the model parameters to reduce search time. However, the output probability computation in (5) makes it difficult to construct a real-time recognition system even with the logarithms if the number of trained HMMs increases. Thus, it is necessary to modify the Viterbi search. That is,

in each frame the HMM states with the scores below a certain threshold are excluded from the evaluation in order to prevent unnecessary output probability calculation. In each frame, the highest score of $\delta_t(i)$ (3) among all the states is obtained and the states within a threshold from the highest score are included in the beam. The transition to the next frame is started only from the states included in the beam, which reduces output probability computation time. Even with the beam, the Viterbi search takes a considerable amount of time. Besides, the beam search can make the recognition accuracy drop by excluding some of the candidates from the evaluation in the early stages.

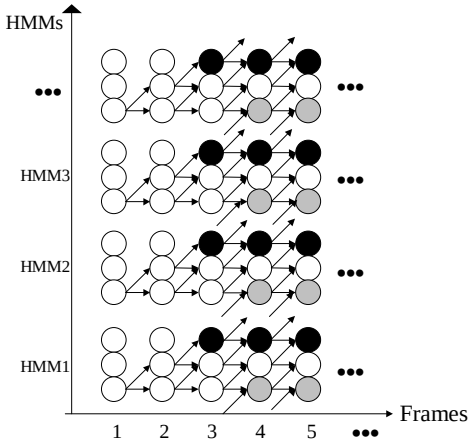

Figure 2. State transition diagram during the parallel Viterbi search

Thus, we propose a new phoneme recognition algorithm by parallelizing the Viterbi beam search. Using this approach, the beam is enlarged as much as possible for better recognition accuracy. Output probability of the states within a beam is computed simultaneously in parallel processors with distributed HMMs while the parallel Viterbi search progresses. Figure 2 shows a transition diagram of the parallel Viterbi search based on the structure of HMMs described in Section 2. The last state of each HMM (indicated by ●) makes transitions to other HMMs in its own processor and HMMs in the other processors. The first state of each HMM (indicated by ○) evaluates the transition probability incoming from other states. Until the 3rd frame, transitions occur within each HMM, from this point on, transitions between different HMMs' occur. Therefore, the parallel phoneme recognition algorithm requires a message passing mechanism between parallel processors.

4. Parallel Phoneme Recognition Algorithm

The parallel phoneme recognition algorithm is implemented on both a root processor and parallel processors. Figure 3 shows the parallel phoneme recognition algorithm. After initializing HMMs transferred from the root processor, parallel processors compute output probabilities with a speech feature vector at every active state in each frame. The root processor gathers transition probabilities of the distributed HMMs and broadcasts the collected transition probabilities of all the HMMs to each processor. Parallel processors set a new beam based on the highest score transferred from the root processor in each frame. Parallel processors send state numbers containing the highest scores in the last frame to the root processor before backtracking. The root processor communicates with parallel processors during backtracking in order to get the best state sequence.

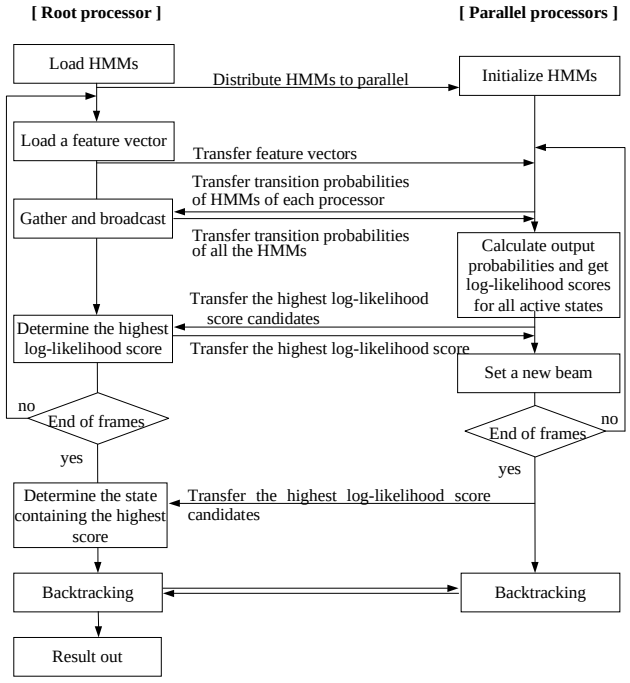

Figure 3. The parallel phoneme recognition algorithm

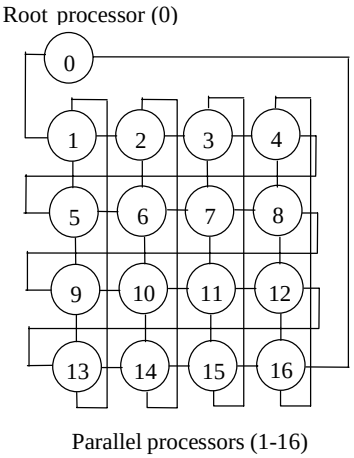

Figure 4. The parallel phoneme recognition machine

In this research, we use a multi-transputer system as a parallel platform. The multi-transputer system is a distributed-memory MIMD machine, which is best suited for the above working mechanism. The prototype machine implemented for the parallel phoneme recognition algorithm consists of 17 processors and 4 hard disks, as shown in Figure 4. Processor 0 is the root processor that works as an interface between the host and the parallel processors. Each processor is a T805 32bit transputer operating at 25MHz. T805 is equipped with four bidirectional links, and communicates with neighboring processors at a speed of 20 Mbits/sec in background. As shown in the figure, the interconnection network is based on a two-dimensional mesh with wrap-around.

5. Experimental Results

To evaluate the performance of the parallel phoneme recognition algorithm, experiments were done under 1, 2, 4, 8, 12, and 16 parallel processors with 48, 720, 2112 phonemes trained from the Korean computer secretary agent DB. Twenty-six MFCC feature vectors extracted from 100 one-second-length test data is used as input features in this experiment.

Our experiments deal with recognition rates, recognition times and speedups of the parallel Viterbi search and the parallel Viterbi beam search. In addition, we show a simple load-balancing method for resolving the workload unbalance during the parallel Viterbi beam search.

5.1 Recognition Rates

To obtain recognition rate, we use %*Correct* and %*Accuracy* shown in (6).

$$\%Correct = \frac{N-S-D}{N} \times 100$$
$$\%Accuracy = \frac{N-S-D-I}{N} \times 100 \qquad (6)$$

N represents the number of total phonemes, *S* represents the number of substitutions, *D* represents the number of deletions, and *I* represents the number of insertions. The *%Correct* and the *%Accuracy* of the parallel phoneme recognition system is shown in Table 1.

	48 CI phonemes	2112 CD phonemes
%Correct	67.30%	83.20%
%Accuracy	49.37%	53.94%

Table 1. Comparison between CI and CD phonemes

As shown in Table 1, the recognition rate of CD phonemes is higher than that of CI phonemes. Although it is difficult to compare the recognition rates of different speech systems, %*Correct* is the range of 60 to 70% in case of English speech data [5].

5.2 Recognition Times and Speedups

Figure 5 and Figure 6 show the recognition times and the speedups of the parallel Viterbi search with and without beam. The recognition times of the parallel speech recognition system are not fast enough because T805 transputers are working at 25MHz. In the case of 720 CD phonemes, the experiments were performed with the configurations of more than 4 processors because there was a memory shortage problem in 1 or 2 processors.

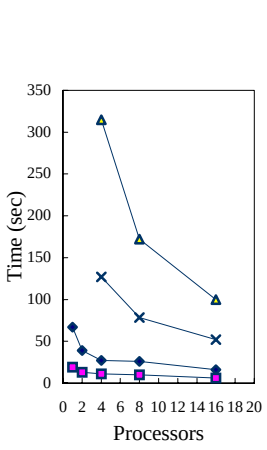

Figure 5. Recognition time

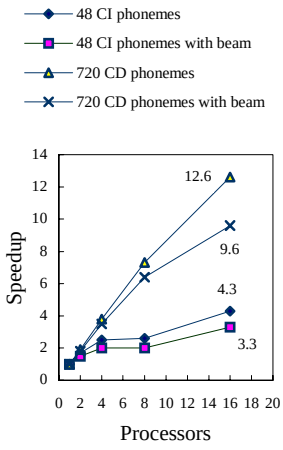

Figure 6. Speedup

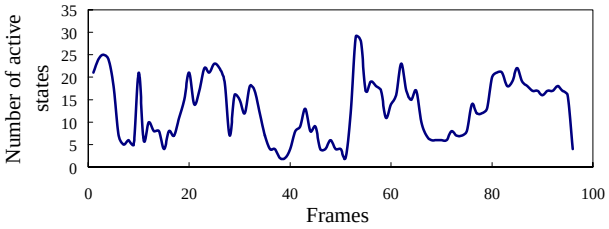

Figure 7. The differences between maximum and minimum number of active states

The speedup of 720 CD phonemes is more significant than that of 48 CI phonemes, because more parallelisms are utilized during the Viterbi search in the case of 720 CD phonemes. When a beam is applied, the recognition time is reduced in both the 48 CI phonemes and the 720 CD phonemes because the unnecessary output probability computations are avoided. The beam was experimentally decided within the threshold which doesn't impact on the recognition rate, however, the existence of beam makes the speedup drop in both cases. This happened primarily due

to the workload unbalances among processors. During the Viterbi beam search, each processor contains different number of active HMM states included in the beam. As a result, each processor has a different amount of loads in calculating output probabilities. The load unbalance problem is shown in Figure 7. The graph shows the differences between maximum and minimum number of active states among 16 processors in evaluating the 2112 CD phonemes.

5.3 Parallel Viterbi beam search with a simple load balancing scheme

The simple load balancing scheme is as follows. 1) Identify processors with larger number of active states (containing 4 more active states than the average) and processors with smaller number of active states (containing 4 less active states than the average). The threshold numbers are decided experimentally. 2) Pair those processors identified in 1) by considering the distance between processors. 3) Redistribute workloads between paired processors.

Figure 8 shows the effect of the simple load balancing scheme. The total recognition time is reduced from 172 seconds to 162 seconds by applying the load balancing scheme. If we apply phonetic constraints and grammar constraints, and increase the number of mixtures and HMM states for better recognition rates, the necessity for load balancing will be more increased. That is because the workload becomes more unbalanced due to the above constraints, and the output probability computation time increases due to the added complexity.

As a future research, we will develop an enhanced load balancing method under these complicated circumstances. The Parsytec CC with 16 PowerPC 604 [14] will be used as a parallel platform.

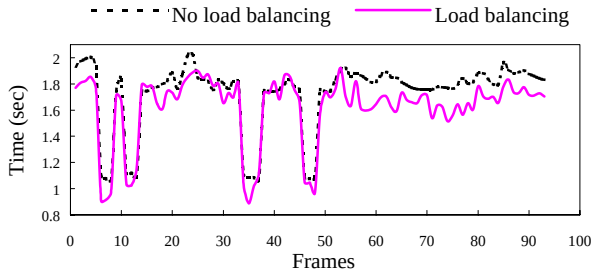

Figure 8. The effect of the simple load balancing (Total recognition time: [No load balancing : 172 seconds], [Load balancing : 162 seconds])

6. Conclusion

We presented a parallel phoneme recognition algorithm based on continuous HMM and implemented it on a MIMD multi-transputer system. Speedup of up to 12.6 was obtained with 720 CI phonemes.

The output probability computation time is considerably reduced by the parallel Viterbi beam search, however the beam search introduced the workload unbalance problem. By applying a simple load balancing technique, an improved recognition time could be obtained. Experimental results show the feasibility of the parallel phoneme recognition algorithm in constructing a real-time parallel speech recognition system based on time-consuming continuous HMM.

References

[1] S.-H. Chung, D.I. Moldovan, and R.F. DeMara, "A Parallel Computational Model for Integrated Speech and Natural Language Understanding", *IEEE Transactions on Computers*, vol. 42, no. 10, pp. 1171-1183, 1993

[2] E.P. Giachin and C.Rullent, "A Parallel Parser for Spoken Natural Language", *Proceedings of IJCAI*, pp. 1537-1542, 1989

[3] G. Huijsen, "Parallel Implementation of Hidden Markov Models on the nCUBE2", M.Sc. thesis, Alparon report, nr. 96-03, Delft University of Technology, 1996

[4] M.Y. Hwand and X.D Huang, "Shared-Distribution Hidden Markov Models for Speech Recognition", *IEEE Trans*. on SAP, vol.l, no.4 pp.414-420, Oct. 1993

[5] K.F. Lee, H.W. Hon, "Speaker-Independent Phone Recognition Using Hidden Markov Models", in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Vol. 37, No 11, Nov. 1989, pp. 1641-1648

[6] K.F.Lee,"Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, no. 4, pp. 599~609, Apr. 1990

[7] K.F.Lee, F. Alleva, "Continuous Speech Recognition", in *Advances in Speech Signal Processing*, pp. 623-650, Marcel Dekker, New York, 1992

[8] Y.J. Lee, Design and Construction of Korean Speech Database, Final report, ETRI, 1996

[9] L. Rabiner and B. H. Juang, Fundamentals of Speech Recognition, Prentice-Hall, New Jersey, 1993

[10] M.D. Riley, A. Ljolje, D. Hindle, and F. Pereira, The AT&T 60,000 word speech-to-text system. In *Proceedings of EUROSPEECH-95*, pages 207-210, 1995

[11] R. M. Schwartz, Y. L. Chow, S.Roucos, M. Krasner, and J. Makhoul, "Improved hidden Markov modeling of phoneme for continuous speech recognition", in *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 1984

[12] F.J. Smith, J. Ming, P. O'Boyle, A.D. Irvine, "A Hidden Markov Model with Optimized Inter-Frame Dependence", *Proc. of ICASSP*, pp. 209-212, 1995

[13] Steven Phillips, Anne Rogers. "Parallel Speech Recognition", In *Preceedings of EUROSPEECH- 97*, pages 135-138, 1997

[14] http://www.parsytec.de/top/products/cc-pp.htm