PARALLEL TRAINING ALGORITHMS FOR CONTINUOUS SPEECH RECOGNITION, IMPLEMENTED IN A MESSAGE PASSING FRAMEWORK

Vladimir Popescu[1, 2], *Corneliu Burileanu*[1], *Monica Rafaila*[1], *Ramona Calimanescu*[1]

[1]Faculty of Electronics, Telecommunications and Information Technology, University "Politehnica" of Bucharest
Bd. Iuliu Maniu 1–3, 7000, Bucharest, Romania
[2]"CLIPS" Laboratory, Institut National Polytechnique de Grenoble
46, Av. Félix Viallet, 38031, Grenoble, France
email: vladimir.popescu@imag.fr

ABSTRACT

A way of improving the performance of continuous speech recognition systems with respect to the training time will be presented. The gain in performance is accomplished using multiprocessor architectures that provide a certain processing redundancy. Several ways to achieve the announced performance gain, without affecting precision, will be pointed out. More specifically, parallel programming features are added to training algorithms for continuous speech recognition systems based on hidden Markov models (HMM). Several parallelizing techniques are analyzed and the most effective ones are taken into consideration. Performance tests, with respect to the size of the training data base and to the convergence factor of the training algorithms, give hints about the pertinence of the use of parallel processing when HMM training is concerned. Finally, further developments in this respect are suggested.

1. INTRODUCTION

It is known that nowadays, though significant progress has been achieved in the microelectronics industry and computers are becoming more and more powerful, large vocabulary continuous speech recognition systems challenge computers when real-time performance is necessary. This is why spoken language technology specialists worldwide are trying to build continuous speech recognition systems and applications with less critical demands with respect to computing resources required. Unfortunately, this relaxation on the resources required involves usually a loss in the precision of modeling and recognition accuracy, through the usage of techniques such as vector quantization (VQ), discrete HMMs (DHMMs), search space pruning.

The alternative approach, taken into consideration by the scientific community [2], is followed here: the usage of parallel computer architectures. More specifically, the training of HMM-based medium vocabulary continuous speech recognition systems is in the focus of our paper. The message passing paradigm for parallel processing has been preferred as a choice of developing and testing framework.

The existence of parallel programming facilities integrated in development environments and of continuous speech recognition applications development baseline platforms, algorithms that add parallel processing facilities to existing speech recognition systems can be designed and *tested*.

However, previous steps have been pursued in parallelizing speech recognition, both at the principial level [9], [10] and at the practical level [3], [4], [8]. Nevertheless, the main focus of these early researches was the actual speech recognition, i.e., the Viterbi *decoding* process [11], given the requirements of real-time speech recognition.

For these reasons, and since performance improvements of the recognition process are being tackled in parallel by our team [6], it is a natural goal to ameliorate the timely performance of the training process for continuous speech recognition. Although several parallel training strategies design hints are provided in the literature [10], [11], no actual parallel training module is available for continuous speech recognition systems. Hence, a research aiming at a feasable parallelizing strategy for continuous speech recognition systems training that should lead to a practical, working parallel training module is motivated by several possible applications. These include, but are not limited to, the following:

- rapid adaptation of a speaker-independent speech recognition system, to the particular voice of a certain speaker, in order to improve recognition accuracy [11]; this can be done either by traditional speaker adaptation methods, such as the Maximum Likelihood Linear Regression (MLLR) [11], or by a partial re-training of the system, to a particular voice [1], the particular application influencing on the choice of the adaptation strategy;
- rapid adaptation of a continuous speech recognition system trained for one language (reach in linguistic resources - such as English), to another language (poorer in terms of linguistic ressources - such as Romanian);

2. CHOICES IN PARALLELIZING TRAINING ALGORITHMS

Since recently the issue of training HMM-based speech recognition systems has not been entirely mitigated, because systems training is basically an offline process for which the real-time requirements are not mandatory. However, with the advent of commercially available large speech databases, large vocabulary speaker-independent continuous speech recognition systems training may become cumbersome in terms of time consumed: it may take tens to hundreds of hours for such systems to be trained in a satisfactory manner.

Moreover, the adaptation of speaker-independent systems to particular speakers or to new languages may take tens of minutes to several hours to complete. Hence, a reduction in training time is a desirable goal that can be achieved through parallel processing, as shown in this paper.

This way, in parallelizing baseline HMM training algorithms, such as the Baum-Welch and Viterbi algorithms [11], several options can be considered:

- *program* parallelism — the parallelization is made with respect to the algorithm itself. In turn, the program parallelism can be achieved in two ways:
  - algorithm-*dependent* — the parallelization depends on the parameter estimation algorithm used (e.g. Baum-Welch or Viterbi) ; a discussion is provided in [7];
  - algorithm-*independent* — the parallelization is independent of the used training algorithm; relevant insights are provided in [10];
- *data* parallelism — the parallelization is made with respect to the data managed by the training algorithm. In turn, the data parallelism can be achieved in several ways:
  - algorithm-*dependent* — the parallelization depends on the parameter estimation algorithm. Taking into account the widespread use of the Baum-Welch algorithm, such a parallelization could be performed with respect to the *training data*; design hints are provided in [11];
  - algorithm-*independent* — the parallelization is independent of the algorithm. Instead, the training data are distributed across several processing units and the algorithm is ran in several instances, for each element of the data partition. In turn, the algorithm-independent data parallelism can be performed in two ways:
    - ∗ with respect to the *training data* — the parametrized speech along with the corresponding labels is spread across several processors and training is performed in parallel, for each element of the data partition;
    - ∗ with respect to the *set of HMMs* (e.g. the phone set of a given language, if phoneme-level modeling is used) — the set of HMMs is partitioned and distributed across several processing units, each unit having the whole training data; the training is performed in parallel, for each group of acoustic models;

Algorithm-dependent program parallelism could solve the linear equations systems giving the model parameters through parallel algorithms, such as parallel Jacobi [7]. However, since iterative processing is alternated with recursions used in computation of the forward and backward probabilities involved in the Baum-Welch algorithm, this manner of parallelization did not lead to performance improvements.

Algorithm-independent program parallelism could be achieved through acoustic vectors partitioning. While the data parallelism involves "space" partitioning (i.e., the training data space is being partitioned), the algorithm-independent program parallelism hinted in [10] implies "time" partitioning (i.e., the acoustic vectors are being split and distributed across several processing units). However, treating each sub-vector independently does not seem to be an appropriate choice for speech recognition systems training since: a) the acoustic vectors are rather short (usually, each feature vector accounts for a few tens of values) and b) the elements within each acoustic vector are strongly correlated. As such, program parallelism may not be well suited for training HMM-based speech recognition systems.

A form of algorithm-dependent data parallelism that explicitly takes into account the Baum-Welch parameter estimation process is suggested in [11]. However, [11] allows only the simulation of parallel processing, needing a human expert that handles the data transfers and accesses. This is why a real parallel algorithm designed in this manner should also take into account the management of data accesses. Thus, such an algorithm can be thought in a master-slave framework, i.e., one processing unit is master, launching the algorithm and commanding several slave processing units, which do not directly interact with each other. The flow of the algorithm is suggested in the message sequence chart (MSC) given in figure 1. The figures illustrating the parallelization strategies are expressed in an Unified Modeling Language (UML)-derived notation; the symbols are UML-specific, except for:

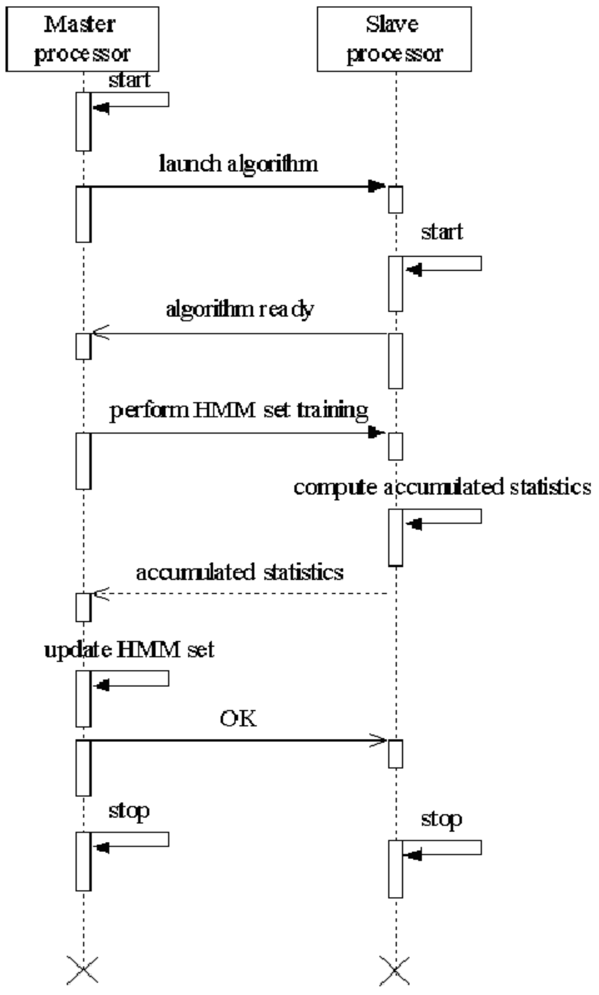

Figure 1: Algorithm-dependent data parallelism

- the continuous full arrows — represent procedure calls;
- the continuous empty arrows — represent control messages;
- the dashed empty arrows — represent data messages;
- the dashed double arrows — represent label positions;
- the circled empty arrows — represent repetitive actions;
- the outgoing continuous empty arrows — represent alternative evolution points;

Basically, the algorithm states that the slave processors compute accumulated statistics using elements of the data being partitioned, and the master (root) processor receives these statistics, combines them and updates the HMM set. The accumulated statistics management works as follows: the slave processing units compute the forward and backward probabilities based on a fraction of the total number of feature vectors, partial sums of functions of these probabilities are computed; finally, the master processor adds these partial sums and performs the appropriate division operations in order to determine the updated model parameters: the transition matrix, the mean vector and covariance (diagonal) matrix of the Gaussians approximating model output. Relevant hints and mathematical details are provided in [11].

A form of algorithm-independent data parallelism with respect to the actual training data involves re-estimation of the HMM set already trained on a subset of the training data, using another subset of the data. In order to provide consistency and practical value to such parallel algorithm, a form of convergence measure can be provided, i.e., a convergence criterion and a distance metric for HMM sets comparison. Detailed insights are given in [7]. As distance measure, an Euclidean-derived distance between the parameters of the models may be chosen [7], in this case the convergence criterion being the minimization of the distance between HMMs. The flow of the algorithm is suggested in the MSC given in figure 2. In this case also, the slave processing units do not interact with each other. Basically, the algorithm states that the slave processors perform actual HMM training on subsets of the speech training data base.

The master processor gathers the estimated models from all the slaves, compares them and, if necessary (the distance is above an empirically-chosen threshold), re-sends the models to the slaves. This way, a slave does not retrain a model that had already been trained with the same data. In order to avoid an avalanche effect (i.e., the excessive multiplication of the HMM sets), a decimation mechanism may be provided: re-send, to a slave processing unit, the HMM set that is the farthest from the HMM set already trained by that slave processor. Alternatively, other decimation mechanisms may be provided, e.g. the random selection of an HMM set to be sent to a slave processor for re-training.

A form of algorithm-dependent data parallelism with respect to the HMM set involves the partitioning of the set of models in a number of subsets equal to the number of processors; the subsets should be as balanced as possible, in terms of component models. Then, the slave processing units train their corresponding subsets using all the training data available. The master processor gathers all the estimated HMM subsets and rebuilds the entire HMM set, containing the updated models. The flow of the algorithm is similar to that illustrated in figure 1; the only difference consists of the object of the exchange: whereas in figure 1 it is the actual data that is being partitioned and the accumulated statistics that are being exchanged, in the latter algorithm it is the HMM set that is being partitioned and the elements of the HMM set that are being exchanged. In our opinion, considering several experiments, from which some reported in [7], out of the parallelization strategies pointed out above, the program-level ones are not appropriate for speech recognition applications, whereas the data-level ones open interesting perspectives. From these latter parallel algorithms, the one dependent on the actual Baum-Welch estimation procedure offers the advantage of being very efficient with respect to the train-

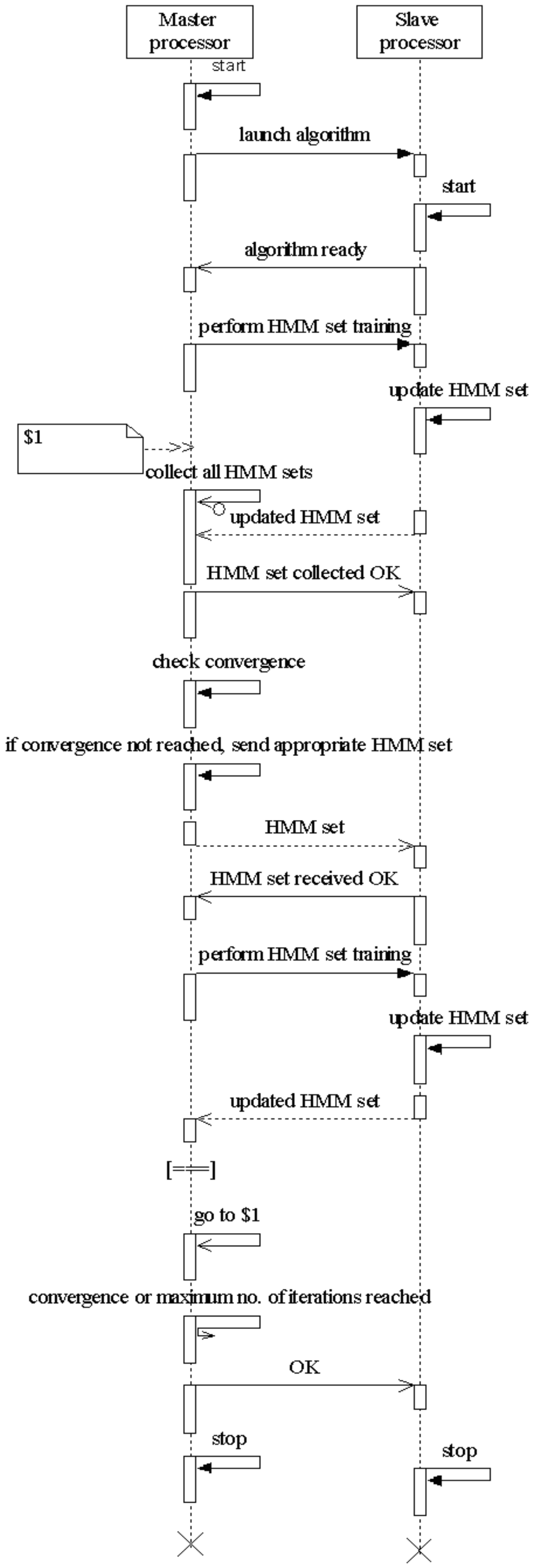

Figure 2: Algorithm-independent data parallelism, with respect to the training data

ing time. The other two, independent of the estimation procedures, offer the advantage of generality: any HMM parameter estimation method could be used in the parallel framework provided.

3. EXPERIMENTAL RESULTS

For the experiments related to the parallelization strategies described in this paper, the following choices have been made: as baseline HMM training system, a subset of the "Hidden Markov Modelling Toolkit" (HTK), provided by Cambridge University [11], has been considered, whereas the parallel framework used was the MPICH implementation of the 'Message Passing Interface" (MPI) standard [5].

The experiments investigate the relationship between the HMM training time in a sequential framework and in a parallel framework consisting of the data-level parallel approaches, versus the amount of training data and the required modeling precision.

The acoustic modeling has been performed at phoneme level, with shared transition matrix (so that the models have exactly the same topology); the 34 phonemes of the Romanian language, along with the short pause, /sp/ and the silence between words, /sil/, were considered. The models are 5-state left-right continuous mixture Gaussian HMMs, the output of each state being modeled through 4 Gaussians. Global model parameter estimation through embedded training is performed [11]. The speech data base used for HMMs training consists of continuously uttered sentences in Romanian language, evenly taken from 10 native Romanian speakers. The sentences use a vocabulary of 3000 words. The characteristics of the database are summarized in table 1.

Size [seconds]	21680
Sample frequency [Hz]	16000
Window size [samples]	600
Pre-emphasis coefficient	0.97
Parametrization	Mel Cepstrum
Number of features / frame	12
Differential features	Delta Cepstra

Table 1: Speech database acoustic characteristics

The training has been performed for an increasing number of speakers, from 1 to 10. The results of the measurements are synthesized in figure 3, for a cluster computer fitted with 3 AMD Athlon processors, out of which one was the master (at 1.7 GHz), and two were the slaves (at 3 GHz). In figure 3, the legend is as follows:

- #1 — no parallelism;
- #2 — algorithm-dependent data parallelism;
- #3 — algorithm-independent data parallelism, with respect to the training data;
- #4 — algorithm-independent data parallelism, with respect to the HMM set.

From the graphs plotted in figure 3, one can observe that the most effective parallelization strategy is the one following the data-level algorithm-dependent approach (notation #2), giving a time twice lower than for the sequential case (notation #1). The data-level algorithm-independent strategy, with respect to the HMM set (notation #4), performs better than the data-level algorithm-independent paralleliza-

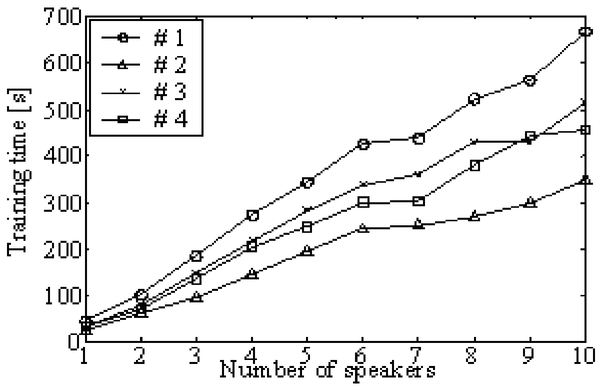

Figure 3: Time measurements for training algorithms, with respect to the size of the training data

tion with respect to the training data (notation #3); the approach #3 gives a reduction in training time, compared to the sequential case (#1), of approximately 25 percent, for the 3-processors parallel computer used. This can be explained by the fact that the parallel algorithm #3 involves the retraining of already estimated acoustic models. The approach #4 offers better results because the groups of HMMs are estimated only once.

Yet, another experiment has been made, which points out a more subtle characterisation of the performances offered by the training algorithms. Thus, the runtime results of the parallel algorithms, as compared to the non parallel ones, can be evaluated with respect to the convergence factor of the Baum-Welch parameter re-estimation algorithm[1]. For reference purposes, we state that in the previous experiment, a convergence factor of 10^{-2} was used.

The experimental setup chosen consisted in a reduced training data base, of 425 seconds of speech material, taken from one speaker and processed according to table 1.

On this data, the algorithm-independent parallelization strategy, with respect to the training data, was chosen, due to its *generality*. This generality could allow for parallelizations of other parameter estimation algorithms, using the aforementioned strategy (e.g. the Viterbi algorithm). More specifically, for this experiment the runtime performance of the non parallel training (notation #1) was compared to the algorithm-dependent parallel training, with respect to the training data (notation #3). The results obtained for this experiments are shown in figure 4, where the notations are the same as in figure 3.

From this figure one can observe that, for a convergence factor superior to 10^{-2} (a "standard" value [11], used in the first experiment also), the non parallel algorithm outperforms the parallel one. This is due to the fact that, for low precision requirements in the estimation of parameters, the interprocessor communication time is important, significantly degrading the performances of the parallel training. However, as the convergence factor lowers (which implies a finer parameter estimation), the actual parameter estimation process becomes important in terms of time consumed, thus yielding better performances for the parallel training algorithm.

[1]The convergence factor is defined as the relative change between successive values of the probability that a given observation sequence had been generated by a certain acoustic model [11].

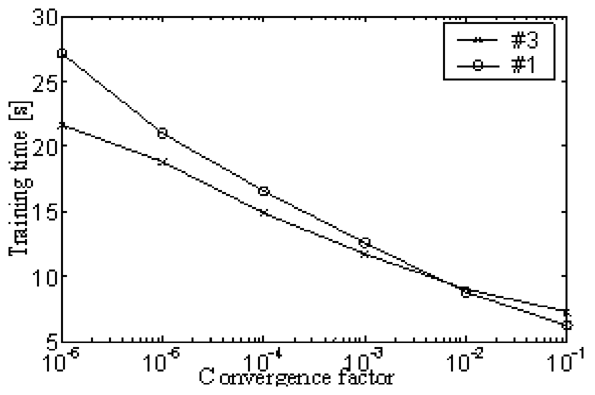

Figure 4: Time measurements for training algorithms, with respect to the convergence factor

Nevertheless, it can be seen (putting alltogether the two experiments) that, even for a convergence factor of 10^{-2}, the parallel algorithm outperforms the sequential one as long as there is an important amount of training data available.

As for the interprocess communication time for the parallelization strategies proposed, informal investigation shows it to be negligible with respect to the computing time for a convergence factor lower than 10^{-2}; however, the interprocess communication time becomes more important for a convergence factor greater than 10^{-2}.

4. CONCLUSIONS AND FURTHER WORK

Although the practical advantages of using parallel processing, when training on large data sets is concerned, are well known, our experiments tried a finer characterisation of various parallelization strategies, that can give valuable hints on which parallel algorithm to use when building large vocabulary speech recognizers. The research described in this paper also tries to bridge the gap between *theoretical* considerations concerning parallel HMM training strategies, and *practical* implementations, that normally involve, as pointed out in [2], considerable logistics, in terms of development and testing time.

One saw that the algorithm-dependent parallelism, with respect to data, offers the best results, but lacks the generality required for an eventual usage on *other* parameter estimation algorithms (e.g. the Viterbi algorithm, commonly used for the initialisation of the HMMs, before the actual Baum-Welch parameter estimation is performed [11]). This is why, in terms of portability, an algorithm-independent parallelization, with respect to data, seems more suited for applications where the development cost is a relevant issue.

Therefore, it can be observed that, when training *precision* is concerned, or there is an important amount of *training data* available, the parallel training is a promising *practical* alternative to the sequential one.

Moreover, since the parallel computer used for the experiments reported in this paper comprises only three processors, using a cluster computer containing several tens or hundreds of processors, an important gain in performance, much significant than that reported in the paper, is expected to be obtained.

A further step in parallelizing the HMM training process may concern the construction of phonetic decision trees, used in order to reinforce the recognition results [11]. It is known that the time for the construction of such a tree, out of the training data, is comparable to the HMM training time itself, or even higher [6]. Hence, a parallel approach in this respect could bring further performance improvements to continuous speech recognition systems.

REFERENCES

[1] C. Burileanu, V. Popescu, "Isolated Word Recognition Engine, Implemented on a Java Platform", in *Proc. Of The 3rd European Conference on Intelligent Systems and Technologies "ECIT" 2004*, Iasi, Romania, 2004.

[2] M. Fleury, A. C. Downton, A. F. Clark, "Parallel Structure in an Integrated Speech Recognition Network", in *Proc. Of The 5th International Euro-Par Conference On Parallel Processing*, Lecture Notes in Computer Science, vol. 1685, pp. 995-1004, Springer, 1999.

[3] O. Kimball, L. Cosell, R. Schwartz, M. Krasner, "Efficient Implementation of Continuous Speech Recognition on a Large Scale Parallel Processor", in *Proc. Of ICASSP 1987*, Dallas, Texas, April 1987.

[4] S. J. Melnikoff, P. B. James-Roksby, S. F. Quigley, and M. J. Russell, "Reconfigurable Computing for Speech Recognition: Preliminary Findings", in *Proc. Of 10th International Conference on Field Programmable Logic and Applications*, Lecture Notes in Computer Science, vol. 1896, pp. 495-504, 2000.

[5] MPI Forum, *MPI: A Message-Passing Interface Standard*, http://www.mpi-forum.org, 2001.

[6] D. Munteanu, E. Oancea, C. Burileanu, "Continuous Speech Recognition Systems Improvements", in *Proc. Of The 3rd Conference on Speech Technology and Human-Computer Dialogue "SpeD2005"*, Romanian Academy Publishing House, Bucharest, 2005.

[7] V. Popescu, C. Burileanu, "Parallel Implementation of Acoustic Training Procedures for Continuous Speech Recognition" , in *Proc. Of The 3rd Conference on Speech Technology and Human-Computer Dialogue "SpeD2005"*, Romanian Academy Publishing House, Bucharest, 2005.

[8] M. K. Ravishankar, "Parallel Implementation of Fast Beam Search for Speaker-Independent Continuous Speech Recognition", *Computer Science & Automation*, Indian Institute of Science, Bangalore, India, 1993.

[9] S. J. Stolfo, Z. Galil, K. McKeown, R. Mills, "Speech Recognition in Parallel", in *Proc. Of The Workshop On Speech And Natural Language, Human Language Technology Conference*, pp. 353-373, Cape Cod, Massachussets, 1989.

[10] W. Turin, "Unidirectional and Parallel Baum-Welch Algorithms", *IEEE Trans. Of Speech and Audio Processing*, IEEE, pp. 516-523, Nov. 1998.

[11] S. Young, G. Evermann, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, Ph. Woodland, *The HTK Book*, Cambridge University, United Kingdom, 2005.