Computational Reduction in Keyword Spotting System Based on the Bucket Box Intersection (BBI) Algorithm

*Kyo-Heok Lee, **Hyung-Soon Kim

Abstract

Evaluating log-likelihood of Gaussian mixture density is major computational burden for the keyword spotting system using continuous HMM. In this paper, we employ the bucket box intersection (BBI) algorithm to reduce the computational complexity of keyword spotting. We make some modification in implementing BBI algorithm in order to increase the discrimination ability among the keyword models. According to our keyword spotting experiments, the modified BBI algorithm reduces 50% of log-likelihood computations without performance degradation, while the original BBI algorithm under the same condition reduces only 30% of log-likelihood computations.

I. Introduction

Keyword spotting is a technique, which detects the predetermined keywords from the unconstrained utterance. It can alleviate the problem of performance limitation in the continuous speech recognition, while enjoying its merit of natural speaking. Therefore, keyword spotting technique has wide range of application areas including voice-activated audiotex service, automatic search of voice message and so on [1].

Most of current keyword spotting system is based on HMM. In general, continuous or semi-continuous HMMs yield better performance than discrete HMM, but they require expensive computational load in evaluating log probability of Gaussian mixture densities.

Several techniques have been proposed to reduce the computations of log-likelihood in continuous HMM. These are the method using rough and detail models [2], tree-based nearest neighbor search method [3], the method using vector quantization [4], bucket Voronoi intersection algorithm [5] and bucket box intersection(BBI) algorithm [6]. Among these techniques, we used BBI algorithm for the computational reduction in the keyword spotting system, because it does not require re-training process and has little additional computation for computational reduction. We made some modification in implementing BBI algorithm for keyword spotting application in order to increase the discrimination ability among the keyword models.

* Hyundai Electronics Industries Co. Ltd.
** Dept. of Electronics Eng., Pusan National University

Manuscript Received : November 9, 1998.

II. Baseline Keyword Spotting System

We implemented a keyword potting system based on continuous HMM. Fig. 1 represents the block diagram of the system. In this system, 12 order LPC cepstra and 12 order LPC delta cepstra are employed as feature vector.

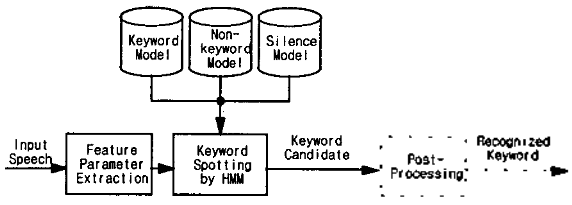


Figure 1. Baseline keyword spotting system.

As recognition unit, we define triphones based on 45 phoneme-like units. Keyword models are constructed by concatenation of these triphones. The HMM topology of phoneme-like unit is represented in Fig. 2. This model consists of 8 transitions and 3 tied distributions of B(egin), M(iddle) and E(nd).

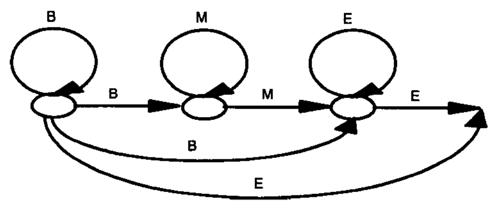


Figure 2. HMM topology of a phoneme-like unit.

The role of non-keyword model in keyword spotting system is to discriminate non-keyword portions from keyword portions in input speech. Therefore the performance of keyword spotting system is heavily dependent on non-keyword models. Too detailed non-keyword models may encroach the keyword portions and too smoothed non-keyword models cannot represent non-keyword portions appropriately.

In this study, we constructed 6 non-keyword models from 45 phoneme-like units by statistical clustering [7]. The distance measure for clustering is

$$D(P_i, P_j) = \sum_{d=1}^{N} D_d(P_i, P_j) \qquad (1)$$

where *pi* and *pj* are *i*-th and *j*-th phoneme-like model, respectively. *N* is the total distribution number of model and $D_d(p_i, p_j)$ is the distance between the distributions of the two phonemes. $D_d(p_i, p_j)$ is given by

$$D_d(p_i, p_j) = \frac{1}{V} \sum_{k=1}^{V} \frac{(\mu_{idk} - \mu_{jdk})^2}{\sigma_{idk}\sigma_{jdk}}, \qquad (2)$$

where V is the dimension of feature vector, and μ_{idk} and σ_{idk} are mean and standard deviation of *d*-th distribution of *i*-th phoneme in dimension *k*, respectively. We used a single silence model.

Overall HMM network consists of keyword, non-keyword, and silence models. In general, null grammar topology is possible because any number of keywords can be included in a sentence. But in this paper, we assumed that only one keyword could be included in input speech. Therefore we constructed overall HMM network as shown in Fig. 3.

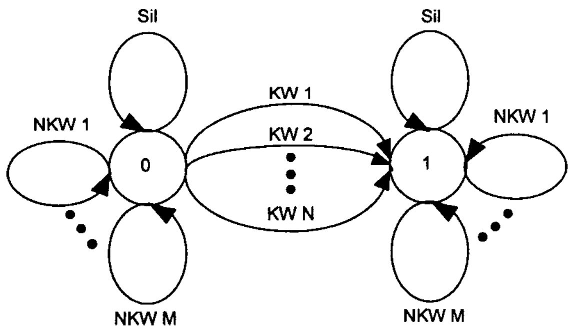


Figure 3. Overall HMM network for keyword spotting.

III. Computational Reduction in Keyword Spotting

3.1. Bucket Box Intersection (BBI) Algorithm [6]

The basic idea of BBI algorithm, like other methods [2]-[5], is to compute log-likelihood only for the Gaussian distributions that reside near the current input feature vector. BBI algorithm substitutes predetermined threshold for log-likelihood of all the other Gaussian distributions that have little effect on observation probability of the current input feature vector.

BBI algorithm constructs a binary tree, which contains the whole Gaussian distributions as its components. Each leafs have several Gaussian distributions which are near to each other. In recognition procedure, BBI algorithm determines the nearest leaf to the current input feature vector, then computes log-likelihood only for the Gaussian distributions in that leaf.

In construction of a tree, BBI algorithm defines the Gaussian boxes for each Gaussian distributions and constructs a binary tree using these Gaussian boxes. A Gaussian box for a Gaussian distribution is defined as the approximation rectangle for the ellipse at a threshold. Projection interval $[a_i, b_i]$ of a Gaussian distribution with a diagonal covariance matrix can be expressed using either an absolute threshold T or a relative threshold R as the following equations:

$$[a_j, b_j] = \mu_i \pm \sqrt{-2\sigma_j \left[T + \frac{1}{2}\log\left((2\pi)^K \prod_{k=1}^{K} \sigma_k{}^2\right)\right]}, \qquad (3)$$
$$T < the\ \max imum\ value\ of\ the\ Gaussian$$

$$[a_j, b_j] = \mu_j \pm \sqrt{-2\sigma_j^2 \log(R)}, \qquad 0 \le R \le 1 \qquad (4)$$

where μ_j and σ_j are mean and standard deviation of the Gaussian distribution in dimension *j*, respectively, and *T* is an absolute threshold and R a relative threshold.

While both of the two thresholds are user-controllable in determining the Gaussian box, the relative threshold enables more accurate modeling of the low probability regions of the mixtures than the absolute threshold, since it considers the width and height of the Gaussian distribution. According to the experiments in the original BBI algorithm, the system using the relative threshold showed better performance than that using the absolute threshold [6], and we also got the same results. Thus we used the relative threshold for determining Gaussian boxes in this paper.

Gaussian box is made up of these projection intervals. For example, a Gaussian box in two dimensional feature space is illustrated in Fig. 4.

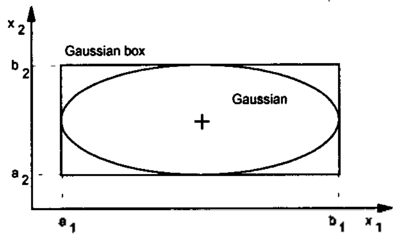


Figure 4. Gaussian box in two dimensional space.

In BBI tree the region belong to a node is called a bucket. A bucket at *j*-th tree depth is divided into two regions by a division hyperplane that is orthogonal to *j*-th axis. A BBI tree with *n* tree depth has 2^n leafs at terminal stage and tree search can be performed by *n* scalar comparison calculations. After determining the nearest leaf to the input feature vector, log-likelihood computations are performed only for the Gaussian distributions in that leaf and predetermined threshold is substituted for log-likelihood of all the other Gaussian distributions. As tree depth and threshold increase, the number of Gaussian distributions belong to a leaf decrease, thus more computational reduction can be achieved at the expense of potential increase in recognition errors. An example of BBI tree in two dimensional space is represented in Fig. 5.

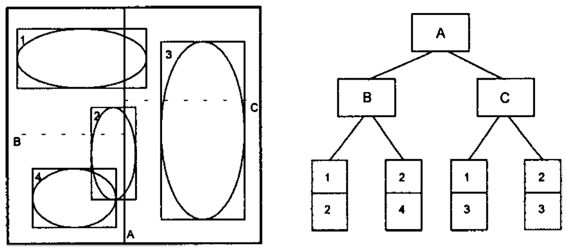


Figure 5. An example of BBI tree in two dimensional feature space.

BBI tree construction procedure is as follows:

Step 1. Find all Gaussian boxes included in current node.

Step 2. For all coordinate axes xi, label the boundaries of all the Gaussian boxes in step 1 as L(ower), U(pper) and sort them along the axis. The hyperplane of current node is determined so that the number of L in left side equals to that of U in right side. Calculate the number of Gaussian boxes, Ci, which intersect with the current hyperplane.

Step 3. The hyperplane, which minimizes Ci is determined as the division hyperplane of current node.

Step 4. Repeat step 1, 2 and 3 until the desired tree depth is achieved.

This method does not involve re-training process and has little additional computation for reduction since only scalar comparisons are required to search the tree.

3.2. Modification of BBI algorithm for keyword spotting applications

According to the BBI reference[6], BBI tree is constructed using all the Gaussian distributions. But in keyword spotting application, BBI tree with all the Gaussian distribution will result in reduced discrimination ability among keywords because of the Gaussian distributions of non-keyword and silence model. The discrimination among keywords is the most important factor in keyword spotting. Thus, we make some modification in implementing BBI tree for keyword spotting applications. At first, BBI tree is constructed using the Gaussian distributions of keyword models only. Then we classify the Gaussian distributions of non-keyword and silence models to each leafs of the pre-constructed BBI tree. The resulting BBI tree will maximize the discrimination ability among keywords.

To minimize approximation error, we carried out the same post-processing scheme as in the original BBI algorithm. A lot of training data are classified into each leafs of the BBI tree, and the contributions of all Gaussian distributions to the classified training data are computed. The Gaussian distributions, which are excluded in the leaf but have large contribution, are assigned to the leaf.

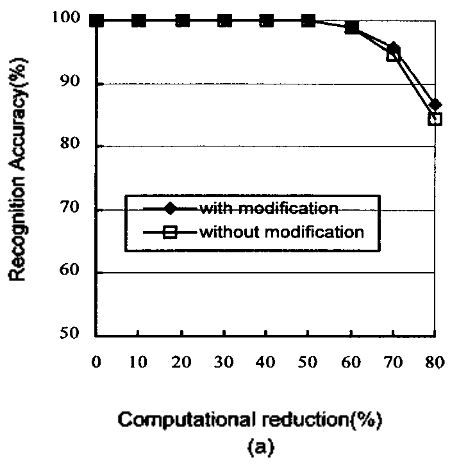
The following experimental results show that the modified BBI tree outperforms the original BBI tree in keyword spotting applications.

IV. Experiments and Results

Our task domain in this paper is the automated attendant service [8]. An incoming call is connected to the desired office division by keyword detection. 6 office division names are selected as keywords. 35 male speakers pronounced training database for keyword model in isolated word. Training database for non-keyword model is phonetically balanced 445 words pronounced by 22 male speakers. 15 male speakers who were not involved in the training database pronounced 6 isolated words and 6 sentences for test.

Speech recognition consists of three steps: feature extraction, computation of observation probability (log-likelihood) and HMM network search. In general, observation probability computation dominates the total computational load. The computational distribution of these three steps in our baseline keyword spotting system is 18%, 64% and 18%, respectively. Therefore, reduction of the computations for the observation probability is essential for reduction of overall recognition time. We carried out keyword spotting experiments using the original and modified BBI algorithms and compared the results.

In Fig. 6, we represented the difference of recognition accuracy between the modified and the unmodified systems. The horizontal axis is the computational reduction in log-likelihood computation compared with the baseline system and the vertical axis is the recognition accuracy. From the figure, it is clear that the modified system yields better performance (50% computational reduction) than the original or unmodified system (30% computational reduction), especially for the embedded keyword case.



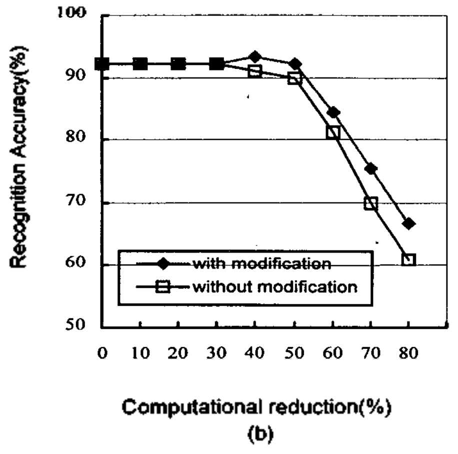
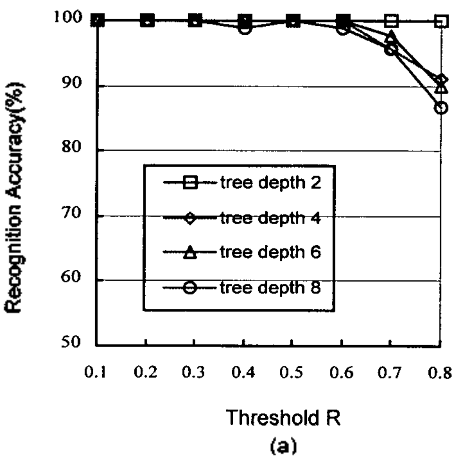


Figure 6. Recognition accuracy according to the threshold and tree depth;
(a) In case of isolated keyword,
(b) In case of embedded keyword.

Figure 7 shows recognition results of the modified system in isolated and embedded word cases according to the threshold and tree depth. As we expected, Fig. 7 shows that as tree depth and threshold increase, which means increased computational reduction, recognition accuracy tends to decrease.



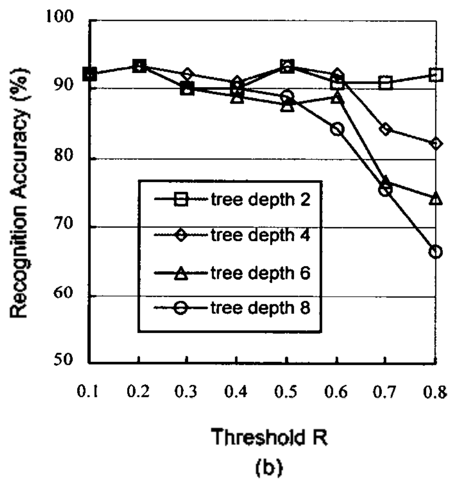


Figure 7. Recognition accuracy according to the threshold and tree depth;
(a) In case of isolated keyword,
(b) In case of embedded keyword.

Fig. 8 shows the direct relationship between recognition accuracy and computational reduction in log-likelihood computation of the modified system as compared with the baseline system. The horizontal axis is the computational reduction as compared with baseline system. It can be seen from the figure that 50% of log-likelihood computation could be reduced, while maintaining the same level of recognition accuracy. In that case, the additional computation due to the BBI tree search was only 0.05% of total recognition time.

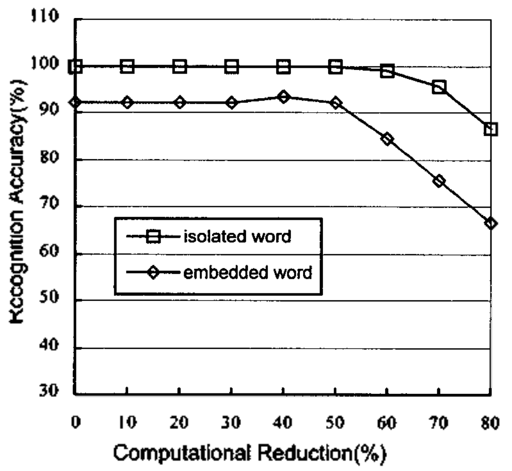


Figure 8. Recognition accuracy according to the computational reduction.

It should be noted that, according to the experiments of original BBI algorithm (reference [6]), the speed-up without performance degradation was about 3 times of baseline system, while in our experiments the speed-up was only 2 times of baseline system. We guess the performance difference is due to the number of Gaussian mixtures in the two experiments. While the number of mixtures in the paper of the original BBI algorithm was about 3300, the number of mixtures in our experiments was only about 200. Therefore we expect that if the number of keywords and consequently the number of Gaussians for keyword models are increased, more than 50% speed-up may be obtained.

V. Conclusion

In this paper, we applied and modified the BBI algorithm to reduce the computational load for continuous HMM based keyword spotting system. Experimental results showed that the modified BBI algorithm reduced 50% of log-likelihood computations without performance degradation, while the original BBI algorithm under the same condition reduced only 30% of log-likelihood computations. It is also noted that the additional computation due to the BBI search was negligible (only 0.05% of total recognition time).

While BBI algorithm is successfully applied and modified to keyword spotting, it has a problem that the approximation error for the input feature vector near the boundary of clusters is relatively large. One approach to deal with this problem was introduced but at the expense of increased memory requirement [9]. We are now investigating a method to reduce the approximation error by adapting the BBI cluster to the input feature vector without additional memory cost.

References

1. H. S. Kim, "Keyword spotting technology," Proceedings of KICS, vol.11, no.9, pp.57-65, Sep. 1994.
2. Y. Komori, M. Yamada, H. Yamamoto and Y. Ohara, "An efficient output probability computation for continuous HMM using rough and detail models," in Proc. EUROSPEECH, pp.1087-1090, 1995.
3. Frank Seide, "Fast likelihood computation for continuous-mixture densities using a tree-based nearest neighbor search," in Proc. EUROSPEECH, pp.1079-1082, 1995.
4. Enrico Bocchieri, "Vector quantization for the efficient computation of continuous density likelihoods," in Proc. IEEE ICASSP, pp.692-695, 1993.
5. J. Fritsch, I. Rogina, T. Sloboda and A. Waibel, "Speeding up the score computation of HMM speech recognizers with the bucket voronoi intersection algorithm," in Proc. EUROSPEECH, pp.1091-1094, 1995.
6. J. Fritsch and I. Rogina, "The bucket box intersection(BBI) algorithm for fast approximative evaluation of diagonal mixture Gaussians," in Proc. IEEE ICASSP, pp.837-840, 1996.
7. H. R. Lee, "A study on the performance improvement of keyword spotting system using phone-based Hidden Makov Models," Master Thesis, Pusan National University, 1996.
8. Y. G. Lee, J. H. Ryu, K. W. Hwang, "The present status in ETRI speech database construction," in Proc. of Korea Speech Comm. & Signal Processing Workshop(KSCSP), pp.265-267, 1995.
9. K. Demuynck, J. Duchateau and D. Van Compernolle, "Reduced Semi-Continuous Models for Large Vocabulary Continuous Speech Recognition in Ductch," in Proc. ICSLP, pp.2289-2292, 1996.

▲ Kyo Heok-Lee

Kyo heok-Lee receieved his B.S. and M.S degree in Electronics engineering from Pusan National University in 1995 and 1997 respectively.
Since 1997 he has been a research engineer at Hyundai Electroics Co., Ltd. Seoul, Korea.
He was involved in research projects including digital audio (MPEG audio, Dolby AC-3, Dolby Prologic, etc.) and speech signal processing.

▲ Hyung Soon-Kim

The Journal of the Acoustical Society of Kerea, Vol. 18, No. 3E, 1999