THE JOURNAL OF THE ACOUSTICAL SOCIETY OF KOREA VOL.24, NO.4E 2005. 12 pp. 123~127

Fast LBG Algorithm to Reduce the Computational Complexity

Dong-Hyun Kim*, Chul-Ho Kang*
*Dept. of Electronics and Communication Engineering, Kwangwoon Univ.
(Received August 19 2005; rivised September 27; accepted November 28 2005)

Abstract

In this paper, we propose a new method for reducing the number of distance calculations in the LBG (Linde, Buzo, Gray) algorithm, which is widely used method to construct a codebook in vector quantization of speech recognition system. The proposed algorithm can reduce the distance calculation between input vector and codeword by utilizing the observation that codewords are quickly stabilized as the number of iteration increases. From the simulation results, it is shown that we can reduce the running times over 43.77% on average in comparison with current LBG algorithm without sacrificing the performance of codebook.

Keywords: *Vector Quantization, Codebook Generation, Clustering Algorithms*

I. Introduction

We consider the codebook generation problem involved in the design of a vector quantizer[1]. The aim is to find code vectors (codebook) for a given set of training vectors (training set) by minimizing the average distance between the training vectors and their representative code vectors (codewords). The vectors are assumed to belong to a K-dimensional Euclidean space. Among several codebook generation algorithms, the LBG (Linde, Buzo, Gray) algorithm is probably the most cited and widely used method[2] because it is easy to implement and produces relatively good codebooks in an efficient manner. The algorithm starts with an initial solution, which is then iteratively improved until the process converges. Each iteration consists of distribution step, where the vectors of the training set are distributed into a set of disjoint clusters, and of codebook upgrade step, where new code vectors are calculated. We introduce a new method for reducing the number of distance calculations in the distribution step without sacrificing codebook performance. The necessity of our paper is that plenty of time is needed to construct a universal codebook in both speech and speaker recognition system. As for speaker recognition system, for example, it takes at least nine hours to construct a codebook per each word, which makes it difficult to implement recognition system. In addition, our fast clustering algorithm can be applied to GMM (Gaussian Mixture Model) based recognition system to reduce complexity.

Our new method records the information of codewords by considering whether they have been changed in the last iteration step or not. According to information of codewords, the clusters are classified into fixed and changing codeword. For training vectors in the fixed clusters, it is unnecessary to search for the nearest clusters among other codeword candidates. Thus, a speed-up can therefore be achieved because the number of fixed clusters is usually high as the number of iteration increases. From the simulation results, we can reduce the comparison search work while maintaining the optimality of codebook. This paper is organized as follows: In Section 2, the conventional LBG is explained and Section 3 describes some existing fast codeword search algorithms, and Section 4 presents the proposed method to reduce the computational complexity of LBG. Simulation results are given in Section 5, and finally, in Section 6 we summarize the conclusion.

Corresponding author: Chul-Ho Kang (chkang5136@kw.ac.kr)
Kwangwoon University, 447-1 Wolgye-Dong, Nowon-Ku, 139-701 Seoul, Korea

II. LBG algorithm

We consider a set of N training vectors T_i in a K-dimensional Euclidean space. The aim is to find a codebook of code vectors by minimizing the average squared distance between the training vectors and their representative codewords. The distance between two vectors is defined by their squared Euclidean distance.

$$d(T_i,C_j) = \| T_i - C_j \|^2 = \sum_{k=1}^{K} (T_{i,k} - C_{j,k})^2 \quad (1)$$

Let be a codebook and the partition of the training set. Under the usual ergodicity assumption, the distortion of the codebook is then determined using the presented training samples as

$$distortion\ (C) = \frac{1}{N} \sum_{i=1}^{N} d(T_i, C_{P_i}) \quad (2)$$

The method starts by generating an initial codebook, which is then improved iteratively using the following two steps. In the distribution step, each training vector is mapped to its nearest codeword in the current codebook.

$$P_i = \arg \min_{1 \le j \le M} d(T_i, C_j) \quad (3)$$

The resulting distribution P is optimal for the given codebook according to (2). In the codebook update step, a new codebook is constructed by calculating the centroids of the partitions

$$C_j' = \frac{\sum_{Pi=j} T_i}{\sum_{Pi=j} 1}, \qquad 1 \le j \le M \quad (4)$$

III. Some existing fast codeword search algorithms

In this section, we introduce previous studies on fast LBG algorithm denoted as partial distortion search (PDS) by Bei and Gray[4], triangular inequality elimination (TIE) by Chen and Tsieh[5]. All these methods reduce the amount of work of the distribution step, in one way or another, by utilizing the distance to the best candidate found so far. The PDS method stops the distance calculation immediately when the partial distance of the new candidate exceeds the minimum distance found so far. The TIE method utilizes the properties of the Euclidean distance function called triangular inequality for eliminating certain codewords from the calculations completely. In general, TIE achieves more calculation reduction than PDS, so TIE is fully adapted as fast codebook searching algorithm. These two algorithms could achieve complexity reduction while maintaining codebook performance. Instead other fast clustering algorithms have been studied while sacrificing little performance[6-8].

IV. The proposed fast LBG algorithm

4.1. Design and concept

It is observed that the LBG makes only local changes in the codebook and the amount of changes differs from codeword to codeword[1][3]. Some codewords will therefore be stabilized within few iterations while others develop much longer before stabilizing. This observation is utilized by detecting fixed codeword indicating whether the codeword was changed during the previous iteration. The codewords are classified into two groups: fixed and changing codeword. The cluster of a fixed codeword is called a fixed cluster, and the cluster of a changing codeword is named as a changing cluster. Through a number of experiments, it is observed that the number of fixed codewords increases with the iteration. The fixed codeword detection is used for reducing the number of distance calculations in the case of

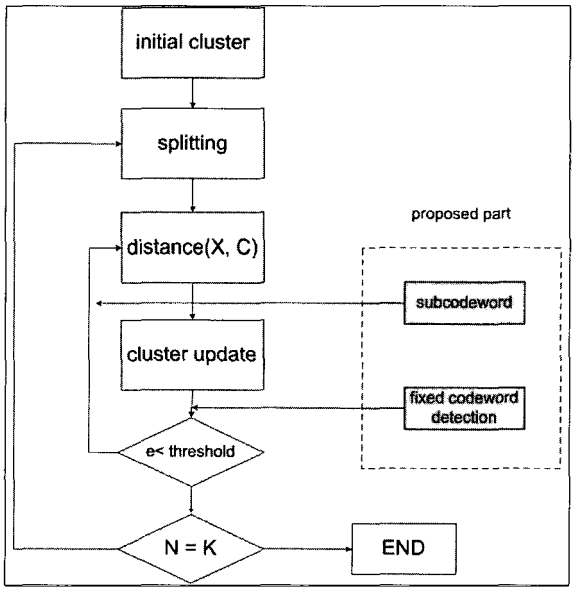

Fig. 1. The proposed fast LBG algorithm.

Table1. The specific amount of calculation.

algorithms	multiplication	addition
Proposed LBG	$\Delta T \Delta CK$	$\Delta T \Delta C(2K-1)$
LBG	TCK	$TC(2K-1)$

training vectors in fixed clusters.

Denote a training vector in a fixed cluster by T_f and a fixed codeword by C_f. And the total number of training vector and codeword are denoted by T,C, respectively with K vector dimension. In this context, it is not necessary to calculate the distance between T_f and other codewords C. And the number of distance calculations in distribution step is $\Delta T \Delta C(2K-1)+\Delta T \Delta CK$. Here, ΔT is $T-T_f$ and ΔC is $C-C_f$. Table 1. indicates the specific amount of calculation. Each component of distance calculation between input vectors and codewords is shown on Table 1. In case fixed codewords are not detected in each iteration, the complexity of proposed method will be the same to that of conventional LBG algorithm. A speed-up is achieved because the number of T_f quickly increases. Furthermore, the number of T_f is proportional to increase in the number of fixed codewords.

Our first observation is that if a fixed codeword was not the nearest codeword for T_f in the previous iteration, it cannot be the nearest codeword in the current iteration either. The partition of the training vectors in a fixed cluster cannot therefore change to another fixed cluster as well as changing clusters. For this reason, it is sufficient to calculate the distances between a changing codeword C_c and a training vector T_c in a changing cluster. Our second observation is that only few number of codeword detected as a fixed codeword is changed to a changing codeword in the later iteration step. The method to deal with this matter is described in detail in the next section. In comparison with conventional LBG, subcodeword and fixed codeword detection are added. The subcodeword is defined to save the previous codeword to perform the fixed codeword detection in the later iteration.

4.2. Fixed codeword detection

Through a number of experiments, it is observed that few fixed codewords are changed to changing codewords in the later iteration. The main reason for this result is that the clustering update of neighbor codewords could cluster input training vector of fixed codewords. To cope with this problem, we define fixed codeword detection, which consists of two searches. Without adapting this method, we can expect to reduce more calculation complexity. However, it is of importance to maintain the performance of codebook with optimality, so this method is needed. The following indicates the steps in fixed codeword detection in detail and this process will be explained more in later chapter.

Step 1 > Construct codeword distance table based on Euclidean distance measure

Step 2 > Classification of fixed and changing codeword according to whether they have been changed or not

Step 2-1 > Calculate radius of newly detected fixed codeword $S=\{C_f;\ f=1,2,\ldots L\}$, $R=\arg\max d(C_f,T_n)$

Step 2-2 > if $d(C_f,C_i)<\alpha R$

C_f is treated as changing codeword

else

C_f is treated as fixed codeword

where $\alpha=\frac{update\ error}{maximum\ error}$ per each iteration, note that $\alpha=1$, until first fixed codeword is detected and update error = codeword in the previous iteration - codeword in the next iteration

Step 3> Generate subcodeword from upgraded clusters and mark fixed codewords

The fixed codeword detection scheme is illustrated in Fig. 2. This scheme is divided into two searches. In case all of the codeword are processed in two searches, the next iteration begins. In the first search, fixed codewords are temporally classified

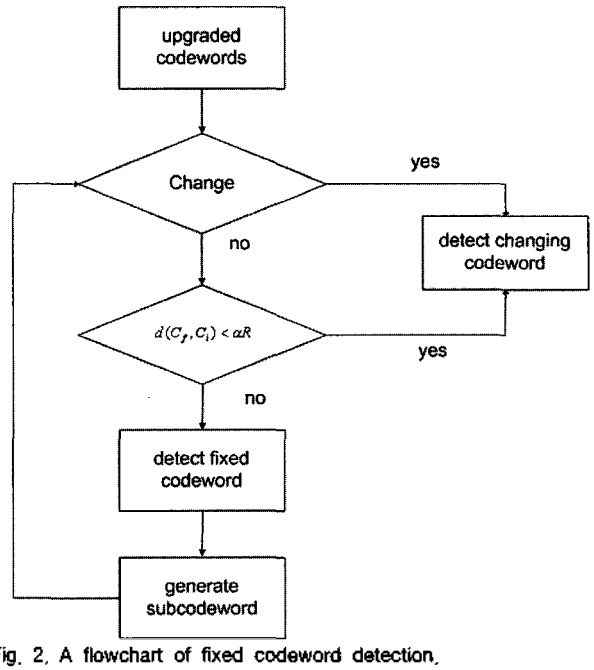

Fig. 2. A flowchart of fixed codeword detection.

according to information whether upgraded codewords are changed or not by comparing upgraded codewords with the previous codeword in the subcodeword to classify fixed codewords. And in the second search, newly detected fixed codewords from the first search are classified into fixed and changing codewords again. Based on the information whether these fixed codewords satisfy the condition or not, we can obtain fixed codeword. This second search is of importance because it can prevent degradation of codebook performance. And then, we mark all of the changing codewords in order to perform distance calculation in the later iteration.

And update error is defined as difference between updated codewords and previous codewords in each iteration. The initial update error value is rather large because initial codeword is generated by random splitting. However, this update error is continually decreased because all codewords are stabilized as the number of iteration increases. And maximum error is maximum value of update error in the iteration where the first fixed codeword is detected. Before this fixed codeword is detected, value is always equal to 1.

V. Experimental setup and results

Test: Database for making world model includes 900 speech data of 300 persons (150 male and 150 female in each age of 20's, 30's) for 4 different words in advance. For the LPC parameter, 20 msec speech is analyzed as a frame and 1/3 frame overlapped. Pre-emphasized speech is converted to 13 LPCC.

The computational decreasing rate of proposed method is 43.77% on average. It is shown that the total codewords are stabilized as the number of iteration increases. Fig. 3. indicates the total computational complexity among three algorithms named TIE, PDS, and proposed method.

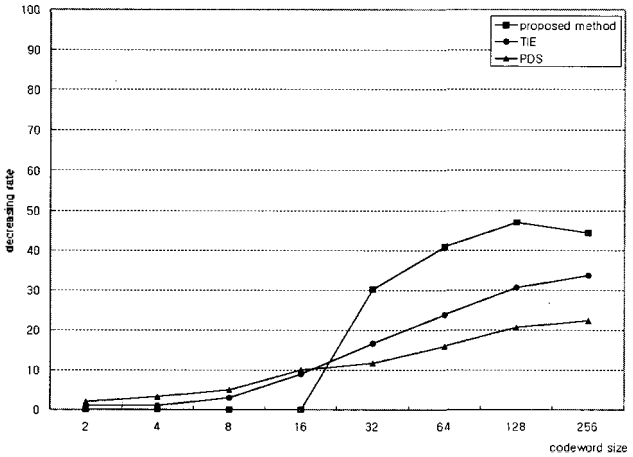

Fig. 3. Total comparison of calculation of proposed fast LBG algorithm.

VI. Conclusion

We proposed a fast LBG algorithm to reduce the computational complexity while preserving the performance of codebook. The key observation is that a large proportion of the distance calculations between codeword and input training vector is unnecessary because only a few codewords remain the changing characteristic to the end of the iterations. An important property of the new method is that the speed-up is based on general property of the LBG algorithm and not on a property of the distance metric such as the triangular inequality. From the simulation results, it is demonstrated that running time of constructing a codebook is reduced by 43.77 %. By adopting fixed codeword detection, the complexity reduction begins between 10 and 15 iterations but it continually increases at the end of iteration. This is because fixed codeword detection is based on the radius parameter derived from stabilization characteristics of codeword in each iteration.

Acknowledgment

The present research has been conducted by the research grant of Kwangwoon University in 2005.

References

1. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*., (Dordrecht, The Netherlands: Kluwer, 1992)
2. Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," IEEE Trans. Commun., 28, 84~95,Jan. 1980.
3. H. Abut, R. Gray and G. Rebolledo, "Vector quantization of speech and speech-like waveforms" , IEEE Trans. Acoust., Speech, Signal Processing, 30 (3), 423~435, Jun 1982
4. C. D. Bei and R. M. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization" , IEEE Trans. Commun., 33, 1132-1133, Oct. 1985.
5. S. H. Chen and W. M. Hsieh, "Fast algorithm for VQ codebook design," Proc. Inst. Elect. Eng., 138, 357~362, Oct. 1991.
6. S. Arya, Mount, D.M, "Algorithms for fast vector quantization" , Data Compression Conference, 30, March-2 April 1993

7. S.-M.Cheng, Lo, K.-T, "Fast clustering process for vector quantisation codebook design" Electronics Letters, 32 (4), 311~312, 15 Feb. 1996
8. Kai Li; Hou-Kuan Huang "Kun-Lun Li, A modified PCM clustering algorithm" , Machine Learning and Cybernetics, 2003 International Conference on 2, 1174~1179 2-5 Nov. 2003

[Profile]

•Dong-Hyun Kim

Dong-Hyun Kim was born in Seoul, Korea, on February 12, 1978. He received the B.S. degree in Electronic Engineering and M.S. degree in Electronics and Communication Engineering from Kwangwoon University, Seoul, Korea, in 2003 and 2005, respectively. Currently, he is working as a research engineer at Mobile Handset Research and Development Center, LG Electronics,Korea. His research interests include speech signal processing and clustering algorithm.

•Chul-Ho Kang

Chul-Ho Kang received the B.S degree in electronics engineering from Hanyang Univ. in 1975. And he also received the M.S and Ph.D. degree in electronics engineering from Seoul National Univ. in 1979 and 1988, respectively. He worked at ADD as a research engineer for five years from 1977 to 1982. He is currently working as a professor since 1983 at the Department of Electronics and Communications Engineering, Kwangwoon University. His research interests include digital signal processing with application to the speech/ speaker recognition and communication systems.