Joint Frame and Gaussian Selection for Text Independent Speaker Verification

Rahim Saeidi[1], *Tomi Kinnunen*[1], *Hamid Reza Sadegh Mohammadi*[2], *Robert Rodman*[3], *Pasi Fränti*[1]
[1]Department of Computer Science and Statistics, University of Joensuu, Finland
[2] Iranian Research Institute for Electrical Engineering, ACECR, Tehran, Iran
[3] Department of Computer Science, North Carolina State University, Raleigh, USA
{rahim,tkinnu,franti}@cs.joensuu.fi, h.sadegh@ijece.org, rodman@ncsu.edu

ABSTRACT

Gaussian selection is a technique applied in the GMM-UBM framework to accelerate score calculation. We have recently introduced a novel Gaussian selection method known as *sorted GMM* (SGMM). SGMM uses scalar-indexing of the universal background model mean vectors to achieve fast search of the top-scoring Gaussians. In the present work we extend this method by using 2-dimensional indexing, which leads to simultaneous frame and Gaussian selection. Our results on the NIST 2002 speaker recognition evaluation corpus indicate that both the 1- and 2-dimensional SGMMs outperform frame decimation and temporal tracking of top-scoring Gaussians by a wide margin (in terms of Gaussian computations relative to GMM-UBM as baseline).

Index Terms**—** speaker verification, Gaussian selection, particle swarm optimization

1. INTRODUCTION

Gaussian mixture model (GMM) is a commonly used statistical speaker modeling technique in text-independent speaker recognition [1]. Usually speaker-dependent GMMs are derived from a speaker-independent universal background model (UBM) by adapting its Gaussian components with maximum a posteriori (MAP) adaptation using speaker's personal training data [2]. This method constructs a natural association between the UBM and the speaker models: for each UBM Gaussian component there is a corresponding adapted component in the adapted speaker model. In the verification phase, each test vector is scored against all the Gaussian components of the UBM, and a small number of the top-scoring components in the corresponding adapted GMM are chosen. The match score is then computed as the log likelihood ratio (LLR) of the speaker GMM and the UBM scores.

Notwithstanding this fast scoring technique, full search of the top-scoring Gaussians from the UBM is still required for each frame. With the typical frame rates - 50 to 100 frames per second - it becomes easily a bottleneck for computations. Reducing the number of UBM mixture evaluations is important on mobile platforms, in the speaker identification task, and when using score normalization techniques including a large number of speaker models to be evaluated, such as Tnorm [3].

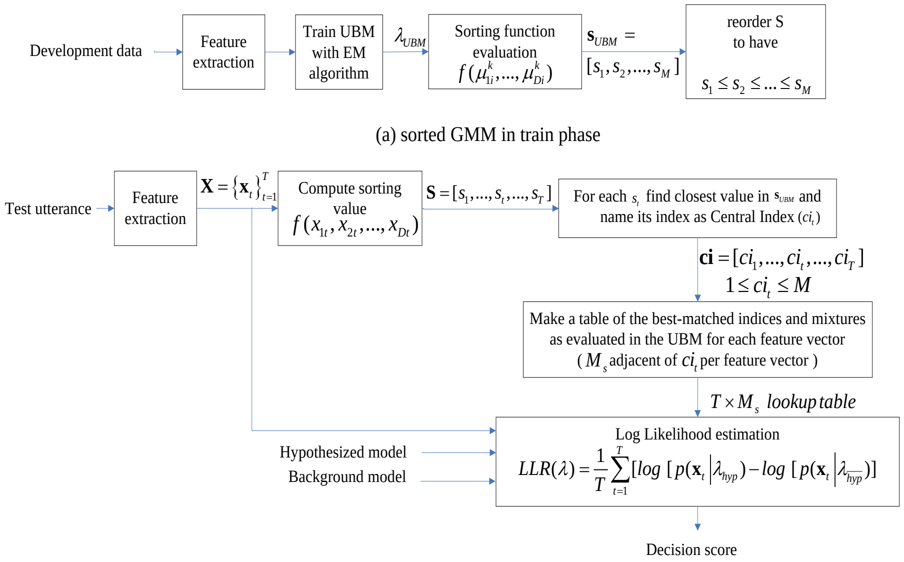
Chan et al. have categorized the existing methods for fast GMM computations in four layers: frame-layer, GMM-layer, Gaussian-layer, and component-layer [4]. Various techniques for reducing the number of GMM computations have been proposed in the literature. Frame decimation [5] reduces the number of frames, whereas hash-modeling [6] and temporal tracking of the top-scoring Gaussians [7] reduce the number of Gaussian density evaluations. Speaker pruning [8], speaker clustering [9] and parametric modeling of the test utterance [10] can also be useful in speaker identification. In this paper our focus, however, will be on the standard likelihood-based matching framework for speaker verification.

The so-called *sorted Gaussian mixture model* (SGMM) algorithm was recently proposed in [11]. SGMM is a novel Gaussian selection method that finds the dominant mixtures from the UBM without extensive search over all Gaussians. This is achieved by using scalar indexing of the UBM mean vectors; in the test phase, each feature vector is projected on the scalar space and the UBM index is used for searching the most-likely top-scoring Gaussians. In [11] the projection plane was optimized by using an evolutionary algorithm [12]. In this paper, we refer this method to as *1-dimensional sorted Gaussian mixture model* (SGMM-1) since it maps high dimensional feature space to scalar values and determines Gaussians to be evaluated using scalar search.

In [11], a speed-up ratio of 15:1 relative to standard GMM-UBM top-scoring [1] was achieved without loss in recognition accuracy. For higher speed-up ratios, say 40:1 to 60:1, the accuracy degrades fast. The reason for this is that projection of high-dimensional vectors onto 1-dimensional space is lossy. In this paper, therefore, we propose an enhanced variant of the SGMM method which addresses this problem. The enhanced variant that we name as the *2-dimensional sorted Gaussian mixture model* (SGMM-2) has two improvements compared to the original formulation [12]. Firstly, it uses a two-dimensional search grid to locate the top-scoring components of the UBM, which leads to more accurate indexing. Secondly, the method simultaneously decides whether a given frame should be passed to Gaussian computations (reduction of frames), meanwhile speeding up the search of the top-scoring components for those frames that passed the frame level test (reduction of Gaussian computations). We demonstrate that the proposed method outperforms frame decimation [5] and temporal tracking [7].

2. SORTED GAUSSIAN MIXTURE MODEL

The sorted Gaussian mixture model is a recently developed method for the fast scoring GMM [11, 12]. To describe SGMM-1 (Fig. 1), assume first that we are given a D-dimensional feature vector $\mathbf{x}_t = [x_{1t}, x_{2t}, ..., x_{Dt}]^T$ related to the speech frame at time t, and a GMM of M Gaussian components. We then define a *sorting key* $s_t = f(x_{1t}, x_{2t}, ..., x_{Dt})$, which is a scalar. The sorting function $f(\cdot)$ is chosen in such a way that neighboring feature vectors provide almost neighboring values of s_t; this allows "ordering" of the D-dimensional feature vectors using the 1-dimensional sorting values and enables efficient indexing technique. The components of the GMM are sorted in ascending order of the associated sorting key according to the vector $\mathbf{S}_{UBM} = [s_1, s_2, ..., s_M]^T$, where $s_1 \le s_2 \le ... \le s_M$.


Development data
Feature extraction
Train UBM with EM algorithm
λ_{UBM}
Sorting function evaluation
$f(\mu_{1i}^{k},...,\mu_{Di}^{k})$
$\mathbf{s}_{UBM} = [s_1, s_2, ..., s_M]$
reorder S to have
$s_1 \le s_2 \le ... \le s_M$
Test utterance
Feature extraction
$\mathbf{X} = \{\mathbf{x}_t\}_{t=1}^{T}$
Compute sorting value
$f(x_{1t}, x_{2t}, ..., x_{Dt})$
$\mathbf{S} = [s_1, ..., s_t, ..., s_T]$
For each s_t find closest value in $\mathbf{s}_{UBM}$ and name its index as Central Index (ci_t)
$\mathbf{ci} = [ci_1, ..., ci_t, ..., ci_T]$
$1 \le ci_t \le M$
Make a table of the best-matched indices and mixtures as evaluated in the UBM for each feature vector
(M_s adjacent of ci_t per feature vector)
$T \times M_s$ lookup table
Hypothesized model
Background model
Log Likelihood estimation
$LLR(\lambda) = \frac{1}{T}\sum_{t=1}^{T}[log\ [p(\mathbf{x}_t|\lambda_{hyp}) - log\ [p(\mathbf{x}_t|\lambda_{\overline{hyp}})]$
Decision score

(a) sorted GMM in train phase

(b) Sorted GMM in test phase

Figure1: *Simplified block diagram of Sorted GMM (a) training phase (b) test phase*

To compute the likelihood of an unknown input feature vector, we first scalar quantize s_t by $\mathbf{S}_{UBM}$ giving s_i, where $1 \le i \le M$. We call this index as the central index. Next, we evaluate the input feature vector's likelihood using the ordinary method by an extensive local search in the neighborhood of the central index i which includes M_s mixtures where $M_s < M$. Only the mixture components with indices in the range $\left[i-k+1 \;\; i+k\right]$ are searched.

Here k is an offset value $(k = M_s / 2)$ and M_s is the search width. In Fig. 1 we summarized the structure of a SGMM-1 system, but exclude the UBM optimization compared to previous work [12] for simplicity. In [12] we used a linearly weighted sorting function defined as follows:

$$s_t = f(\mathbf{x}_t) = \langle \mathbf{a}, \mathbf{x}_t \rangle = \sum_{i=1}^{D} a_i x_{it} \qquad (1)$$

Here x_{it} stands for ***i***th MFCC in the *t*th feature vector and the weight vector $\mathbf{a}$ is optimized using a so-called *particle swarm optimization* (PSO) algorithm [13]. Considering only UBM and one GMM memory storage requirement, the memory storage required for the SGMM-1 is $(2D+2)/(2D+1)$ times that needed for the GMM-UBM assuming diagonal covariance matrices. The negligible extra storage is required to store the sorting key quantization table. On the other hand, the number of Gaussian computations, a measure of speed-up, is reduced to $M_s + C$ over $M + C$ in the baseline GMM-UBM system. Here C is the number of top-scoring mixtures whose corresponding mixtures are evaluated in the speaker GMM. Thus, the speed-up factor of the SGMM-1 algorithm is approximately $(M+C)/(M_s+C)$. We have ignored the computations required for finding the M_s Gaussians since it is negligible compared to Gaussian component evaluations. To incorporate Tnorm [3] in the score calculation, *N* additional Tnorm impostor speakers need to be considered, leading to speed-up factor of SGMM-1 as $(M+NC)/(M_s+NC)$. Like any Gaussian selection algorithm operating in Gaussian layer, if large cohort sets used for Tnorm score normalization, the speed-up factor tends to unity.

2.1. Optimization of the Sorting Function

Assume a sequence of feature vectors denoted as $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$. We can write this sequence as $\mathbf{X} = [\mathbf{c}_1^T, ..., \mathbf{c}_{D/3}^T, \Delta\mathbf{c}_1^T, ..., \Delta\mathbf{c}_{D/3}^T, \Delta\Delta\mathbf{c}_1^T, ..., \Delta\Delta\mathbf{c}_{D/3}^T]^T$ where the superscript T stands for matrix transpose and the vectors are composed of MFCCs and their delta and double delta parameters, each subset with D/3 dimensions. We rewrite the feature vectors as $\mathbf{X} = \left(\mathbf{x}_1'^T, \ldots, \mathbf{x}_D'^T\right)$ where the $\mathbf{x}_i'$s represents the MFCCs for $1 \le i \le D/3$, ΔMFCCs for $D/3 < i \le 2D/3$ and ΔΔMFCCs for $2D/3 < i \le D$ over all feature vectors.

The introduction of a sorting function as the sum of feature vector elements in [5] was based on correlations between sorting function values $\mathbf{s} = [s_1, \ldots, s_T]$ and $\mathbf{x}_i'$s where they are highly correlated for low index values such as $\mathbf{x}_1'$, $\mathbf{x}_2'$ and $\mathbf{x}_3'$ with correlations decreasing to $\mathbf{x}_D'$. In general, a sorting function which generates sorting values highly correlated with $\mathbf{x}_i'$s provides better results in the Gaussian selection stage. In the linearly weighted form of the sorting function as in the SGMM-1 after having UBM trained, the adjustable weights $\mathbf{a}$ can be learned in a data-dependent manner from UBM training material. The fitness function is function of the weights $\mathbf{a}$ and in [12] we defined it as follows:

$$Fitness(\mathbf{a}) = \sum_{i=1}^{D} \frac{E\{(\mathbf{x}_i' - E(\mathbf{x}_i'))(\mathbf{s}' - E(\mathbf{s}'))\}}{\sqrt{E\{(\mathbf{x}_i' - E(\mathbf{x}_i'))^2\}E\{(\mathbf{s}' - E(\mathbf{s}'))^2\}}} \qquad (2)$$

Here $E\{.\}$ stands for mathematical expectation. The weights $\mathbf{a}$ should be chosen in such a way that the fitness function (2) is maximized. The optimization problem defined so far has the goal of discovering the optimal weights for the sorting function, $\mathbf{a}$ so as to attain the maximum correlations.

We expect that, by maximizing the fitness function (2) yields weights of the sorting function (1) so that neighboring feature vectors would result in almost-neighboring sorting values. These would consequently correspond to the most valuable mixtures for this purpose since they provide a level of discrimination comparable to top-C selection in conventional GMM-UBM systems. Because the search space is unknown, an optimization algorithm is needed that is capable of searching a wide area while avoiding local maximums. For this purpose we have selected the PSO algorithm [12].

3. ENHANCED SORTED GAUSSIAN MIXTURE MODEL

In the current work we extend the concept from 1-dimension to 2-dimensions by utilizing two sorting functions as:

$$s_1^t = f_1(\mathbf{x}_t) = \langle \mathbf{a}, \mathbf{x}_t \rangle$$
$$s_2^t = f_2(\mathbf{x}_t) = \langle \mathbf{b}, \mathbf{x}_t \rangle \qquad (3)$$

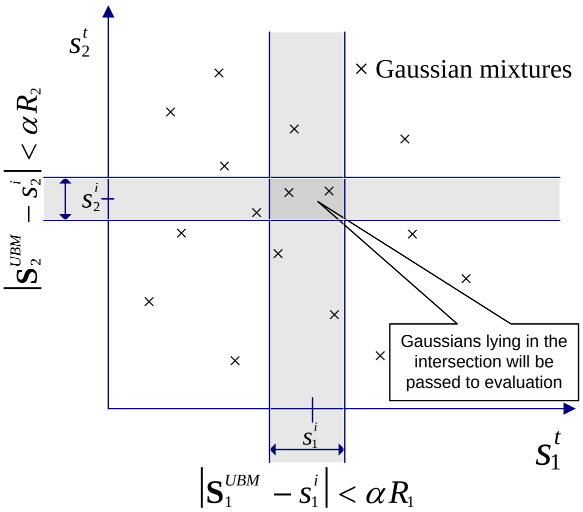

s_2^t
× Gaussian mixtures
$\left|\mathbf{S}_2^{UBM} - s_2^i\right| < \alpha R_2$
s_2^i
Gaussians lying in the intersection will be passed to evaluation
s_1^i
s_1^t
$\left|\mathbf{S}_1^{UBM} - s_1^i\right| < \alpha R_1$

Figure 2: *Illustration of SGMM-2 algorithm*

Here $\mathbf{a}$ and $\mathbf{b}$ are designed to be close to orthogonal and optimized using the PSO algorithm [13]. We alter the way that the mixtures are considered for evaluation by focusing on those mixtures whose sorting values are in the set of specified adjacent values taken over the central mixture sorting values. Thus we may find the mixtures to be evaluated by considering those mixtures whose corresponding sorting values in two dimensions exist in the rectangular neighborhood specified by $\left|\mathbf{S}_1^{UBM} - s_1^i\right| < \alpha R_1$ and $\left|\mathbf{S}_2^{UBM} - s_2^i\right| < \alpha R_2$, where α is a control parameter to specify the neighborhood. $\mathbf{S}_1^{UBM}$ and $\mathbf{S}_2^{UBM}$ are the sorting values of UBM means according to $f_1(.)$ and $f_2(.)$, respectively R_1 and R_2 are the range of $\mathbf{S}_1^{UBM}$ and $\mathbf{S}_2^{UBM}$ accordingly, while s_1^i and s_2^i are scalar-quantized values of the unknown input feature vector sorting values, s_1^t and s_2^t, by $\mathbf{S}_1^{UBM}$ and $\mathbf{S}_2^{UBM}$, respectively. In other words, mixture components located in the intersecting area of the search areas specified by $f_1(.)$ and $f_2(.)$ go through the Gaussian evaluation process, and if there is no mixture in this area (this situation happens usually for low values of α), that feature vector is dropped from Gaussian evaluation. Finding the intersection area is shown in Fig. 2 for a simple 16 mixture case where their sorting keys are plotted as (×) in projection plane.

The memory storage required for the SGMM-2 is $(2D+3)/(2D+1)$ times that needed for the GMM-UBM. Based on the fact that some of the frames may be dropped from the Gaussian evaluation process, speed-up rate differs from one test segment to another and therefore the average value over all test segments must be considered for comparison with SGMM-1. In addition, in SGMM-2 the number of Gaussians to be evaluated for each frame, M_s^t, may be less than the C mixtures to be evaluated in UBM and GMM, hence we define a more efficient parameter C^t to be the lesser number of the top-scoring mixtures as:

$$C^t = \begin{cases} C & M_s^t \geq C \\ M_s^t & M_s^t < C \end{cases} \tag{4}$$

Thus, the speed-up factor (without considering the minor overheads due to SGMM) of the SGMM algorithm for N_{test} test segments, each of them with T_n frames, can be computed as:

$$1/N_{test} \sum_{n=1}^{N_{test}} (T_n(M+C)) / (\sum_{t=1}^{T_n} M_s^t + C^t)$$

$$1/N_{test} \sum_{n=1}^{N_{test}} (T_n(M+NC)) / (\sum_{t=1}^{T_n} M_s^t + NC^t) \quad \text{with Tnorm}$$

Considering the fact that SGMM-2 algorithm works also in frame level speed-up, when using large cohort sets for Tnorm, speed-up factor will not fall down dramatically compared to SGMM-1.

3.1 Optimizing weights in two dimensional case

For the new 2-dimensional sorted GMM concept, PSO must optimize two weigh vectors denoted as $\mathbf{a}$ and $\mathbf{b}$. Considering the definition of 2-dimensional sorted GMM in (2) we propose a new fitness function as,

$$Fitness(\mathbf{a},\mathbf{b}) = Fitness(\mathbf{a}) + Fitness(\mathbf{b}) - abs(\mathbf{a}.\mathbf{b} / |\mathbf{a}||\mathbf{b}|) \tag{5}$$

where Fitness(.) is defined as in (2). The last term is the absolute value of the cosine of the angle between the two vectors. The absolute value accounts for vectors that are directionally opposed (at an angle between π/2 and 3π/2). The term is subtracted because the function is maximized when the vectors are orthogonal, i.e. the cosine is zero. By allowing PSO to find these two weighting vectors we will be able to come up with a two-dimensional search in sorted GMM space.

4. PERFORMANCE EVALUATION

4.1 Experimental Setup

The speaker recognition experiments were conducted on the NIST 2002 speaker recognition corpus [14], which consist of cellular telephone conversational speech and excerpts from the Switchboard corpus. Making use of MFCCs we followed the same configuration as described in [12] to utilize NIST 2002 and 2001 SRE data for constructing UBM and speaker models. The UBM model order is set to 1024 throughout the experiments. The evaluation of the speaker verification system is based on detection error trade-off (DET) curves, which show the tradeoffs between false alarm (FA) and false rejection (FR) errors. We also used the minimum detection cost function (MinDCF) and equal error rate (EER) [9] to measure accuracy. EER is defined as the point where FA and FR errors are equal, and MinDCF is a weighted sum of FA and FR where false acceptance are punished more. Fixed rate decimation [5] and top-C scoring mixture tracking technique [7] are included in the comparisons as well.

4.2 Experiments and Results

We compare the performance of 2-dimensional PSO-optimized SGMM while considering the standard GMM-UBM system as the baseline [2]. The detection error trade-off (DET) plots for the proposed method are summarized in Fig. 3. In SGMM-2 the control parameter, α was chosen as 2 %, 3 %, 5 %, 10 %, 15 % and 20 % which gives average speed-up ratios of 157:1, 85:1, 37:1, 11:1, 5:1 and 3:1, respectively. Frame decimation algorithm [5] simply chooses one over every *N* (decimation factor) frames. Mixture tracking [7] algorithm first builds a look-up table for each mixture where most probable mixtures to be selected after this mixture are listed. In utterance scoring after first frame whole search in UBM, top scoring mixture selected and a subset size of mixtures in its list selected for next frame evaluation. Every 100 frames, a full search is performed to avoid “dead-end transitions”. The effect of Tnorm score normalization [3] can also be seen in Fig 3. Figure 4 presents EER versus speed-up factor (relative to standard top-scoring in a GMM-UBM system) for SGMM-1, SGMM-2, decimation and mixture tracking algorithms. It can be seen that SGMM-2 outperforms SGMM-1, decimation and mixture tracking, the two latter ones by a wide margin.

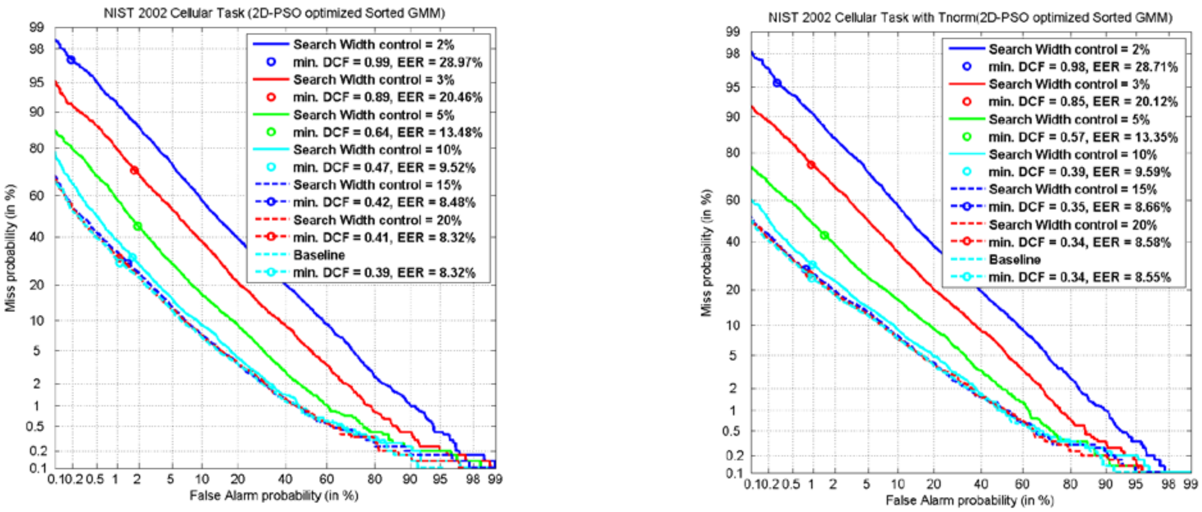


Figure 3: DET curves for SGMM-2 algorithm without (left) and with (right) test normalization (Tnorm).

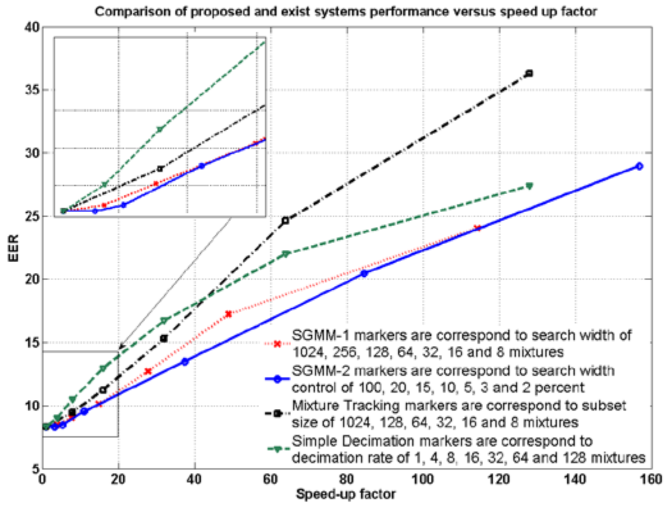


Figure 4: Algorithms Comparison in the space of EER versus speed-up factor (relative to standard top-scoring in GMM-UBM).

5. CONCLUSIONS

We have introduced a 2-dimensional sorted GMM for computationally efficient speaker recognition. Our experiments indicate the effectiveness of the proposed method over the 1-dimensional version. The SGMM algorithm also performs much better than two well-known methods, decimation and temporal mixture tracking algorithms.

6. ACKNOWLEDGEMENT

The work of first author was supported by a grant from the Centre for International Mobility (CIMO).

7. REFERENCES

[1] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture speaker models," *IEEE Trans. on Speech Audio Processing*, vol. 3, no. 1, pp. 72-83, Jan. 1995.

[2] D. A. Reynolds, T. F. Quatieri, and R B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19-41, Jan. 2000.

[3] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Processing*, vol. 10, pp. 42-54, 2000.

[4] A. Chan, R. Mosur, A. Rudnicky, and J. Sherwani, "Four-layer categorization scheme of fast GMM computation techniques in large vocabulary continuous speech recognition systems," in *Proc. INTERSPEECH'04*, pp. 689-692, 2004.

[5] J. McLaughlin, D. A. Reynolds and T. Gleason, "A study of computation speed-ups of the GMM-UBM speaker recognition system," in Proc. Eurospeech '99, pp. 1215-1218, 1999.

[6] R. Auckenthaler and J. Mason, "Gaussian selection applied to text-independent speaker verification, "in Proc."A Speaker Odyssey," Speaker Recognition Workshop, 2001, pp.83–86.

[7] B. Tydlitat, J. Navratil, J. W. Pelecanos, G. N. Ramaswamy, "Text-Independent Speaker Verification in Embedded Environments", in *Proc. ICASSP'07*, vol. 4, pp. 293-296, 2007.

[8] T. Kinnunen, E. Karpov, and P. Fränti, "Real time speaker identification and verification," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 14, no 1, pp. 227-288, Jan. 2006.

[9] V. R. Apsingekar and P. L. De Leon, "Speaker Model Clustering for Efficient Speaker Identification in Large Population Applications," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 17, no. 4, pp. 848 - 853, May 2009.

[10] H. Aronowitz, D. Burshtein, "Efficient Speaker Recognition Using Approximated Cross Entropy (ACE)", *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 15, no 7, pp. 2033-2043, Sep. 2007.

[11] H. R. Sadegh Mohammadi and R. Saeidi, "Efficient implementation of GMM based speaker verification using sorted Gaussian mixture model," in *Proc. EUSIPCO'06*, Florence, Italy, Sept. 4-8, 2006.

[12] R. Saeidi, H. R. Sadegh Mohammadi, T. Ganchev, R. D. Rodman, "Particle Swarm Optimization for Sorted Adapted Gaussian Mixture Models," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 17, no 2, pp. 344-353, Feb. 2009.

[13] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. on Neural Network*s, vol. IV, pp. 1942-1948, 1995.

[14] The NIST Year 2002 Speaker Recognition Evaluation, http://www.nist.gov/speech/tests/