Real-Time Speaker Identification and Verification

Tomi Kinnunen, Evgeny Karpov, and Pasi Fränti

***Abstract*—In speaker identification, most of the computation originates from the distance or likelihood computations between the feature vectors of the unknown speaker and the models in the database. The identification time depends on the number of feature vectors, their dimensionality, the complexity of the speaker models and the number of speakers. In this paper, we concentrate on optimizing vector quantization (VQ) based speaker identification. We reduce the number of test vectors by pre-quantizing the test sequence prior to matching, and the number of speakers by pruning out unlikely speakers during the identification process. The best variants are then generalized to Gaussian mixture model (GMM) based modeling. We apply the algorithms also to efficient cohort set search for score normalization in speaker verification. We obtain a speed-up factor of 16:1 in the case of VQ-based modeling with minor degradation in the identification accuracy, and 34:1 in the case of GMM-based modeling. An equal error rate of 7% can be reached in 0.84 s on average when the length of test utterance is 30.4 s.**

***Index Terms*—Gaussian mixture model (GMM), pre-quantization, real-time, speaker pruning, speaker recognition, vector quantization (VQ).**

I. Introduction

SPEAKER RECOGNITION refers to two different tasks: *speaker identification* (SI) and *speaker verification* (SV) [1]–[3]. In the identification task, an unknown speaker X is compared against a database of known speakers, and the best matching speaker is given as the identification result. The verification task consists of making a decision whether a voice sample was produced by a claimed person.

A. Motivation

Applications of speaker *verification* can be found in biometric person authentication such as an additional identity check during credit card payments over the Internet. The potential applications of speaker identification can be found in multi-user systems. For instance, in *speaker tracking* the task is to locate the segments of given speaker(s) in an audio stream [4]–[7]. It has potential applications in automatic segmentation of teleconferences and helping in the transcription of courtroom discussions.

Speaker identification could be used in *adaptive user interfaces*. For instance, a car shared by many people of the same family/community could recognize the driver by his/her voice, and tune the radio to his/her favorite channel. This particular application concept belongs to the more general group of *speaker adaption methods* that are already employed in speech recognition systems [8], [9]. Speaker-specific codecs in *personal speech coding* have been also demonstrated to give smaller bit rates as opposed to a universal speaker-independent codec [10].

Speaker identification have also been applied to the verification problem in [11], where the following simple rank-based verification method was proposed. For the unknown speaker's voice sample, K nearest speakers are searched from the database. If the claimed speaker is among the K best speakers, the speaker is accepted and otherwise rejected. Similar verification strategy is also used in [12].

Speaker identification and adaptation have potentially more applications than verification, which is mostly limited to security systems. However, the verification problem is still much more studied, which might be due to (1) lack of applications concepts for the identification problem, (2) increase in the expected error with growing population size [13], and (3) very high computational cost. Regarding the identification accuracy, it is not always necessary to know the exact speaker identity but the *speaker class* of the current speaker is sufficient (speaker adaptation). However, this has to be performed in real-time. In this paper, we focus on decreasing the computational load of identification while attempting to keep the recognition accuracy reasonably high.

B. Review of Computational Speed-Up Methods

A large number of methods have been proposed for speeding up the *verification* process. Specifically, Gaussian mixture model (GMM) based verification systems [14], [15] have received much attention, since they are considered as the state-of-the-art method for text-independent recognition. Usually, speaker-dependent GMMs are derived from a speaker-independent *universal background model* (UBM) by adapting the UBM components with *maximum a posteriori* (MAP) adaptation using each speaker's personal training data [15]. This method incudes a natural hierarchy between the UBM and the personal speaker models; for each UBM Gaussian component, there is a corresponding adapted component in the speaker's personal GMM. In the verification phase, each test vector is scored against all UBM Gaussian components, and a small number (typically 5) of the best scoring components in the corresponding speaker-dependent GMMs are scored. This procedure effectively reduces the amount of needed density computations.

In addition to the basic UBM/GMM approach, a number of other hierarchical methods have been considered for GMM. Beigi *et al.* [12] propose a hierarchical structuring of the speaker database with the following merging strategy. Two

Manuscript received November 21, 2003; revised August 20, 2004. The work of E. Karpov was supported by the National Technology Agency of Finland (TEKES) under Project "Puhetekniikan uudet menetelmät ja sovellukset," TEKES dnro 8713103. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Geoffrey Zweig.

The authors are with the Department of Computer Science, University of Joensuu, Joensuu FIN-80101, Finland (e-mail: Tomi.Kinnunen@cs.joensuu.fi).

Digital Object Identifier 10.1109/TSA.2005.853206

closest GMMs are merged, and the process is repeated until the number of GMMs is 1. A similar approach using the *ISODATA* clustering algorithm has been recently proposed by Sun *et al.* [16] for the identification task. They report identification accuracy close to full search with speed-up factors from 3:1 to 6:1. The relative speed-up of their algorithm was higher for increased number of speakers.

Auckenthaler and Mason [17] applied UBM-like *hash model*, in which for each Gaussian component, there is a shortlist of indices of the expected best scoring components for each individual GMM. Using the shortlist of the hash model, only the corresponding components in the individual GMM are then scored. By increasing the lengths of the shortlists, scores can be computed more accurately, but with an increased computational overhead. Auckenthaler and Mason reported a speed-up factor of about 10:1 with a minor degradation in the verification performance.

McLaughlin *et al.* [18] have studied two simple speed-up methods for the GMM/UBM-based verification system: (1) decreasing the UBM size, and (2) decimating the sequence of test vectors with three simple methods. They noticed that the UBM could be reduced by a factor of 4, and the test sequence up to a factor of about as high as 20 without affecting the verification performance. McLaughlin *et al.* [18] state (p. 1218):

> "What is surprising is the degree to which feature vectors can be decimated without loss in accuracy. ... The key factor seems to be the acoustic variety of the vectors scored, not the absolute number of vectors."

However, they did not experiment the combination of decimation and reduced UBM.

An efficient GMM-based speaker identification system has also been presented by Pellom and Hansen [19]. Since the adjacent feature vectors are correlated and the order of the vectors does not affect the final score, the vector sequence can be reordered so that nonadjacent feature vectors are scored first. After the scoring, worst scoring speakers are pruned out using a *beam search* technique where the beam width is updated during processing. Then, a more detailed sampling of the sequence follows. The process is repeated as long as there are unpruned speakers or input data left, and then the best scoring speaker is selected as the winner. Pellom and Hansen reported speed-up factor of 6:1 relative to the baseline beam search.

Recently, more advanced hierarchical models have been proposed for efficient speaker verification [20], [21]. Xiang and Berger [20] construct a tree structure for the UBM. Multilevel MAP adaptation is then used for generating the speaker-specific GMMs with a tree structure. In the verification phase, the target speaker scores and the UBM scores are combined using an MLP neural network. Xiang and Berger reported a speed-up factor of 17:1 with a 5% relative increase in the EER. They also compared their method with the hash model of Auckenthaler and Mason [17]. Although the method of Xiang and Berger gave slightly better verification accuracy (from EER of 13.9% to EER of 13.5%) and speed-up (from 15:1 to 17:1) as compared to the hash GMM, the Xiang's and Berger's method is considerably more complex than the hash GMM.

C. Contributions of This Study

The literary review herein shows that most of the speed optimizations have been done on GMM-based systems. In this study, we optimize *vector quantization* (VQ) based speaker recognition, because it is straightforward to implement, and according to our experiments, it yields equally good or better identification performance than the baseline GMM based on maximum likelihood training using the EM algorithm.

Most of the computation time in VQ-based speaker identification consists of distance computations between the unknown speaker's feature vectors and the models of the speakers enrolled in the system database. *Speaker pruning* [19], [22], [23] can be used to reduce the search space by dropping out unlikely speakers "on the fly" as more speech data arrives. We survey and compare several speaker pruning variants. We also propose a new speaker pruning variant called *confidence-based speaker pruning*. The idea is to wait for more speech data until we are confident to decide whether a certain speaker could be safely pruned out.

We optimize the other components of the recognition system as well. We reduce the number of test sequence vectors by silence removal and pre-quantization, and show how the pre-quantization methods can be combined with the speaker pruning for more efficient identification. A *vantage-point tree* (VPT) [24] is used for indexing the speakers' code vectors for speeding up the nearest neighbor search. Our main contribution is a systematic comparison and combining of several optimization methods.

Although the framework presented in this study is built around VQ-based speaker modeling, the methods are expected to generalize to other modeling paradigms. We demonstrate this by applying the best pre-quantization and pruning variants to GMM-based identification.

Finally, we demonstrate that the methods apply also to the verification task. Pre-quantization is applied for searching a *cohort set* online for the client speaker during the verification process, based on the closeness to the input vectors. We propose a novel cohort normalization method called *fast cohort scoring* (FCS) which decreases both the verification time and the equal error rate.

The rest of the paper is organized as follows. In Section II, we review the baseline speaker identification, and consider the computational complexity issue in more detail, focusing on the real-time processing in general level. A detailed description of the speaker pruning algorithms follows then in Section III. In Section IV, we utilize the speed-up methods to the verification problem. Section V describes the experimental setup. Test results with discussion are given in Section VI, and conclusions are drawn in Section VII.

II. VQ-Based Speaker Identification

A. General Structure

The components of a typical VQ-based speaker identification [25]–[28] system are shown in Fig. 1. *Feature extraction* transforms the raw signal into a sequence of 10- to 20-dimensional feature vectors with the rate of 70–100 frames per second. Commonly used features include *mel-cepstrum* (MFCC) and *LPC-*

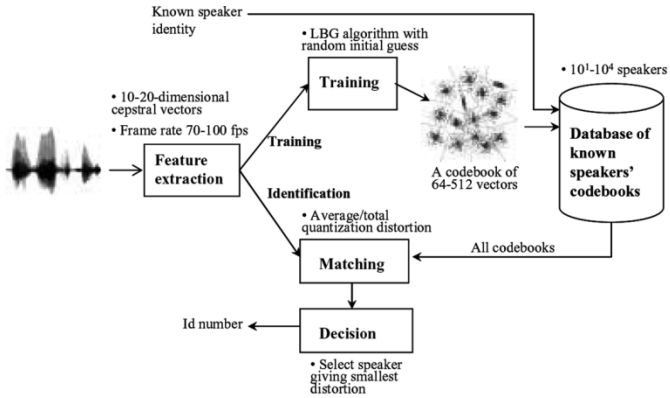


Fig. 1. Typical VQ-based closed set speaker identification system.

cepstrum (LPCC) [29], [30]. They measure short-term spectral envelope, which correlates with the physiology of the vocal tract.

In the training phase, a speaker model is created by clustering the training feature vectors into disjoint groups by a clustering algorithm. The *LBG algorithm* [31] is widely used due to its efficiency and simple implementation. However, other clustering methods can also be considered; a comparative study can be found in [32]. The result of clustering is a set of M vectors, $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_M\}$, called a *codebook* of the speaker.

In the identification phase, unknown speaker's feature vectors are matched with the models stored in the system database. A *match score* is assigned to every speaker. Finally, a 1-out-of-N decision is made. In a closed-set system this consists of selecting the speaker that yields the smallest distortion.

The match score between the unknown speaker's feature vectors $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$ and a given codebook $C = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M\}$ is computed as the *average quantization distortion* [25]

$$D_{\text{avg}}(X, C) = \frac{1}{T} \sum_{i=1}^{T} e(\boldsymbol{x}_i, C) \tag{1}$$

where $e(\boldsymbol{x}_i, C) = \min_{\boldsymbol{c}_j \in C} ||\boldsymbol{x}_i - \boldsymbol{c}_j||^2$, and $||\cdot||$ denotes the Euclidean norm. Several modifications have been proposed to the baseline VQ distortion matching [27], [33]–[37].

B. Time Complexity of Identification

In order to optimize speaker identification for real-time processing, first the dominating factors have to be recognized. In order to compute $D_{\text{avg}}(X, C)$, the nearest neighbors of each $\boldsymbol{x}_i \in X$ from the codebook C are needed. With a simple linear search this requires $O(TM)$ distance calculations. Computation of the squared Euclidean distance between two d-dimensional vectors, in turn, takes d multiplications and $d - 1$ additions. Therefore, the total number of floating point operations (flops) for computing $D_{\text{avg}}(X, C)$ is $O(TMd)$. The computation of $D_{\text{avg}}(X, C)$ is repeated for all N speakers, so the total identification time is $O(NTMd)$.

The efficiency of the feature extraction depends on the selected signal parametrization. Suppose that the extraction of one vector takes $O(f)$ flops. The total number of flops for feature extraction is then $O(Tf)$, where T is the number of vectors. Notice that the feature extraction needs to be done only once. To sum up, total number of flops in identification is $O(Tf + NTMd) = O(T(f + NMd))$. The standard signal processing methods (MFCC, LPCC) themselves are very efficient. By assuming $f \ll NMd$, we can approximate the overall time as $O(TNMd)$.

The dimensionality d is much smaller than N, M, and T. For instance, about 10–20 mel-cepstral coefficients is usually enough due the fast decay of the higher coefficients [29]. There is no reason to use a high number of cepstral coefficients unless they are properly normalized; the coefficients with a small magnitude do not contribute to the distance values much.

C. Reducing the Computation Time

The dominating factors of the total identification time are the number of speakers (N), the number of vectors in the test sequence (T), and the codebook sizes (M). We reduce the number of speakers by pruning out unlikely speakers during the matching, and the number of vectors by silence removal and by pre-quantizing the input sequence to a smaller number of representative vectors prior to matching. In order to speed up the nearest neighbor search of the codebooks, we utilize *vantage-point trees* (VPT) [24] for indexing the code vectors in the models. VPT is a balanced binary search tree where each node represents a code vector. In the best case (fully balanced binary tree), the search takes $O(\log_2 M)$ distance computations. Unfortunately, the VPT as well as other indexing structures [38] apply only to metric distance functions. Since (1) does not satisfy the triangular inequality, we can index only the code vectors but not the codebooks themselves.

D. Real-Time Speaker Identification

The proposed system architecture is depicted in Fig. 2. The input stream is processed in short buffers. The audio data in the buffer divided into frames, which are then passed through a simple energy-based silence detector in order to drop out non-information bearing frames. For the remaining frames, feature extraction is performed. The feature vectors are pre-quantized to a smaller number of vectors, which are compared against *active speakers* in the database. After the match scores for each speaker have been obtained, a number of speakers are pruned out so that they are not included anymore in the matching on the next iteration. The process is repeated until there is no more input data, or there is only one speaker left in the list of active speakers.

E. Pre-Quantization

In *pre-quantization* (PQ), we replace the original test vector sequence X by a new sequence $\hat{X}$ so that $|\hat{X}| < |X|$. In order to gain time, the total time spent for the PQ and matching must be less than the matching time without PQ. The motivation for using PQ is that, in practice, the adjacent feature vectors are close to each other in the feature space because of the gradual movements of the articulators. McLaughlin *et al.* [18] applied three simple PQ methods prior to GMM matching, and reported that the test sequence could be compressed by a factor of 20:1 without compromizing the verification accuracy. This clearly suggests that there is a lot of redundancy in the feature vectors.

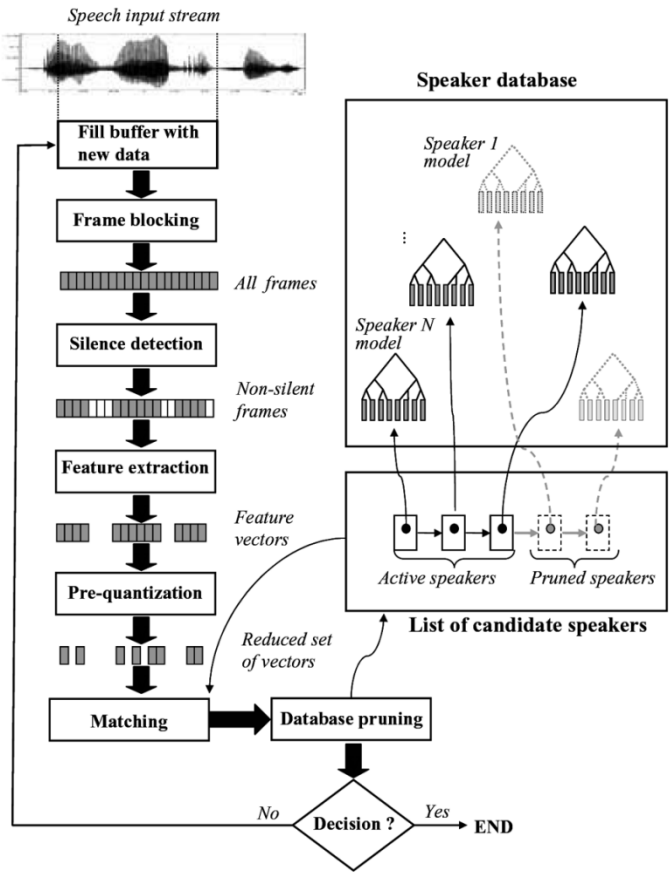


Fig. 2. Diagram of the real-time identification system.

We consider four different pre-quantization techniques: 1) *random subsampling*, 2) *averaging*, 3) *decimation*, and 4) *clustering-based PQ*. In random subsampling and averaging, the input buffer is processed in nonoverlapping segments of $M > 1$ vectors. In random subsampling,each segment is represented by a random vector from the segment. In averaging, the representative vector is the centroid (mean vector) of the segment. In decimation, we simply take every Mth vector of the test sequence, which corresponds to performing feature extraction with a smaller frame rate. In clustering-based PQ, we partition the sequence X into M clusters using the LBG clustering algorithm.

III. Speaker Pruning

The idea of speaker pruning [19], [22], [23] is illustrated in Fig. 3. We must decide how many new (nonsilent) vectors are read into the buffer before next pruning step. We call this the *pruning interval*. We also need to define the *pruning criterion*.

Fig. 4 shows an example how the quantization distortion (1) develops with time. The bold line represents the correct speaker. In the beginning, the match scores oscillate, and when more vectors are processed, the distortions tend to stabilize around the expected values of the individual distances because of the averaging in (1). Another important observation is that a small

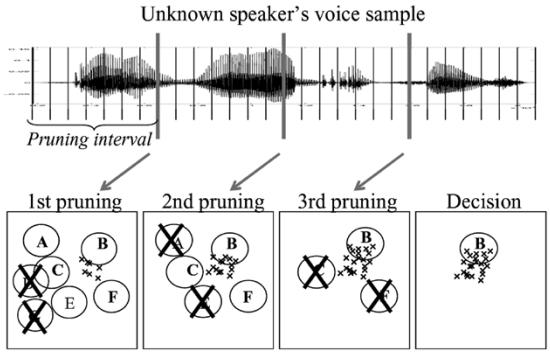


Fig. 3. Illustration of speaker pruning (pruning interval = 7 vectors).

amount of feature vectors is enough to rule out most of the speakers from the set of candidates.

We consider next the following simple pruning variants: *static pruning* [23], *hierarchical pruning* [22], and *adaptive pruning* [23]. We also propose a novel pruning variant called *confidence-based pruning*. The variants differ in their pruning criteria.

The following notations will be used.

X	processing buffer for new vectors;
A	indices of the active speakers;
C_i	codebook of speaker i;
N	size of the speaker database.

A. Static Pruning (SP)

The idea is to maintain an ordered list of the best matching speakers. At each iteration, M new vectors are read in, match scores of the active speakers are updated, and K worst matching speakers are pruned out (Algorithm 1). The update of the match scores can be done efficiently by using cumulative counts of the scores. The control parameters of the method are M and K. Fig. 3 gives an example of the method with parameters $M = 7$ and $K = 2$.

Algorithm 1 Static Pruning (SP)

$A := \{1, 2, \ldots, N\}$; $X := \emptyset$;
while ($|A| > 1$) **and** (speech data left) **do**
  Insert M new vectors into buffer X ;
  Update $D_{avg}(X, C_i)$ for all $i \in A$;
  Prune out K worst speakers from A ;
end while
Decision: $i^* = \arg\min_i \{D(X, C_i) | i \in A\}$;

B. Hierarchical Pruning (HP)

For each speaker i, two codebooks are stored in the database: a *coarse* and a *detail* codebook, denoted here as C_i^c and C_i^d, respectively. Both codebooks are generated from the same training data, but the coarse codebook is much smaller than the detail one: $|C_i^c| \ll |C_i^d|$. First, K worst speakers are pruned out by matching the vectors against the coarse models. Scores of the remaining models are then recomputed using the detail models (Algorithm 2). The control parameters of the method are the the sizes of the codebooks and K.

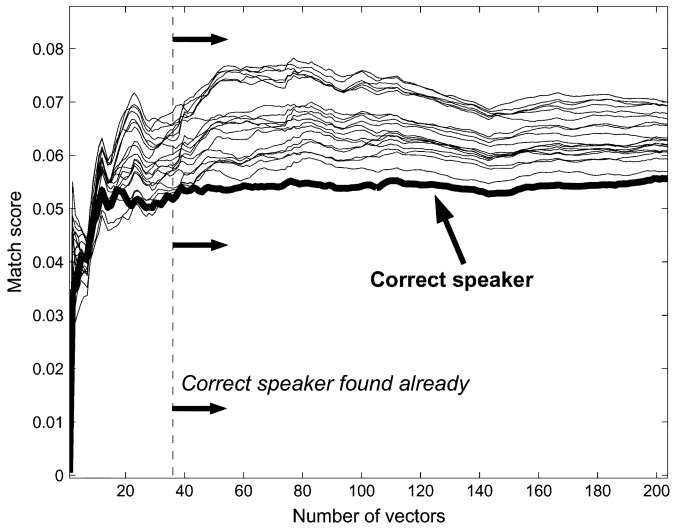
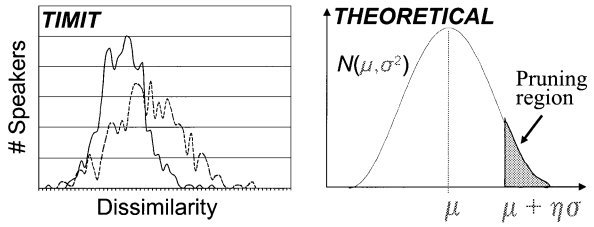


Fig. 4. Illustration of match score saturation ($N = 20$ speakers from the TIMIT corpus).



(a) (b)

Fig. 5. (a) Match score distributions from the TIMIT corpus. (b) Illustration of the pruning threshold.

Algorithm 2 Hierarchical Pruning (HP)

Let $C_c = \{C_1^c, \ldots, C_N^c\}$ be the coarse models ;
Let $C_d = \{C_1^d, \ldots, C_N^d\}$ be the detail models ;
$A := \{1, 2, \ldots, N\}$;
Read the whole test sequence into buffer X ;
Compute $D_{avg}(X, C_i^c)$ for all $i \in A$;
Prune out K worst speakers from A ;
Compute $D_{avg}(X, C_i^d)$ for all $i \in A$;
Decision: $i^* = \arg\min_i \{D_{avg}(X, C_i^d) | i \in A\}$;

C. Adaptive Pruning (AP)

Instead of pruning a fixed number of speakers, a pruning threshold Θ based on the distribution of the scores is computed, and the speakers whose score exceeds this are pruned out (see Algorithm 3). The pruning threshold Θ is computed as

$$\Theta = \mu_D + \eta \cdot \sigma_D \tag{2}$$

where μ_D and σ_D are the mean and the standard deviation of the active speakers' match scores, and η is a control parameter. The larger η is, the less speakers are pruned out, and vice versa. The formula (2) has the following interpretation. Assuming that the match scores follow a Gaussian distribution, the pruning threshold corresponds a certain *confidence interval* of the normal distribution, and η specifies its width. For $\eta = 1$, the speakers above the 68% confidence interval of the match score distribution will be pruned out; that is approximately $(100 - 68)/2 = 16\%$ of the speakers. This interpretation is illustrated in the right panel of Fig. 5. We have found out experimentally that the Gaussian assumption holds sufficiently well in practice. The left panel of Fig. 5 shows two real score distributions computed from two different subsets of the TIMIT corpus [39].

Algorithm 3 Adaptive Pruning (AP)

$A := \{1, 2, \ldots, N\}$; $X := \emptyset$;
while ($|A| > 1$) **and** (speech data left) **do**
  Insert M new vectors into buffer X ;
  Update $D_{avg}(X, C_i)$ for all $i \in A$;
  Update Pruning threshold Θ ;
  Prune out speaker i if $D_{avg}(X, C_i) > \Theta$;
end while
Decision: $i^* = \arg\min_i \{D_{avg}(X, C_i) | i \in A\}$;

Notice that the mean and variance of the score distribution can be updated efficiently using the running values for these. Since the unlikely speakers (large scores) are pruned out iteratively, the variance of the match scores decreases with time. The control parameters of the method are M and η.

D. Confidence-Based Pruning (CP)

In confidence-based pruning, only speakers whose match scores have stabilized are considered for pruning. If the match score is poor but it still oscillates, the speaker can still change its rank and become the winner. Thus, we remove only speakers that have already stabilized and whose match score is below a given threshold. This is illustrated in Fig. 6, in which the speakers are at given one per line, and the time (vector count) increases from left to right. The numbers in the cells show the match scores, gray color indicates that the speaker has stabilized, and black indicates that the speaker has been pruned out. Notice that both the stabilization and pruning can happen in the same iteration.

The pseudocode of the method is given in Algorithm 4. Two score values are maintained for each active speaker i: the one from the previous iteration ($D_{\text{prev}}[i]$), and the one from the current iteration ($D_{\text{curr}}[i]$). When these two are close enough to each other, we mark the speaker as stabilized. Stabilized speakers are then checked against the pruning threshold as defined in (2). There are three adjustable parameters: the pruning interval (M), the stabilization threshold (ϵ) and the pruning threshold control parameter (η).

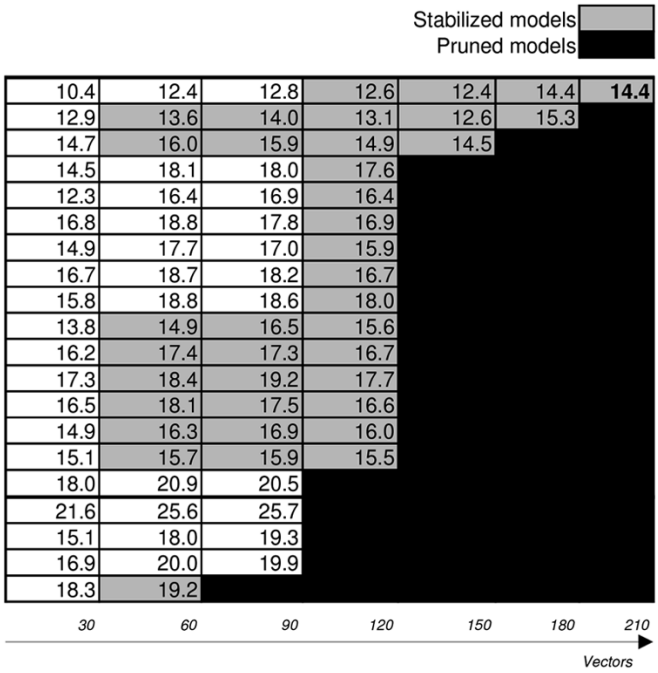


Fig. 6. Illustration of the confidence-based pruning.

Algorithm 4 Confidence-Based Pruning (CP)

$A := \{1, 2, \ldots, N\}$; $X := \emptyset$;
for $i := 1, \ldots, N$ **do**
  $D_{prev}[i] := 0$; stable[i] := **false** ;
end for
while ($|A| > 1$) **and** (speech data left) **do**
  Insert M new vectors into buffer X ;
  Update $D_{avg}(X, C_i)$ for all $i \in A$;
  Update pruning threshold Θ ;
  for $i \in A$ **do**
    $D_{curr}[i] := D_{avg}(X, C_i)$;
  end for
  for $i \in A$ **do**
    if ($|1 - D_{prev}[i]/D_{curr}[i]| < \epsilon$) **then**
      stable[i] = **true** ;
    end if
    if (stable[i]) **and** ($D_{curr}(X, C_i) > \Theta$) **then**
      Prune out speaker i from A ;
    else
      $D_{prev}[i] := D_{curr}[i]$;
    end if
  end for
end while
Decision: $i^* = \arg\min_i \{D_{avg}(X, C_i) | i \in A\}$;

E. Combining PQ and Pruning (PQP)

Pre-quantization and pruning can be combined. Algorithm 5 combines clustering-based PQ and static pruning. First, the whole input data is pre-quantized by the LBG algorithm [31]. Using the match scores for the quantized data, K worst scoring speakers are pruned out, and the final decision is based on comparing the unquantized data with the remaining speaker models. We refer the ratio of the number of pruned speakers to the number of all speakers as the *pruning rate*.

Algorithm 5 PQ + Static Pruning (PQP)

$A := \{1, 2, \ldots, N\}$;
Read new data into buffer X ;
$\hat{X} :=$ LBG-Clustering(X, M)
Compute $D_{avg}(\hat{X}, C_i)$ for all $i \in A$;
Prune out K worst speakers from A ;
Compute $D_{avg}(X, C_i)$ for all $i \in A$;
Decision: $i^* = \arg\min_i \{D_{avg}(X, C_i) | i \in A\}$;

IV. Efficient Cohort Scoring for Verification

In this Section, we apply pre-quantization for speeding up the scoring in the verification task. Current state-of-the-art speaker verification systems use the Bayesian likelihood ratio [40] for normalizing the match scores [41], [42]. The purpose of the normalization is to reduce the effects of undesirable variation that arise from mismatch between the input and training utterances.

Given an identity claim that speaker S produced the vectors $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$, two likelihoods $p(X \mid S)$ and $p(X \mid \bar{S})$ are estimated. The former presents the likelihood that speaker S produced X (*null hypothesis*), and the latter presents the likelihood that X was produced by someone else (*alternative hypothesis*). The two likelihoods are combined using the log-likelihood ratio [1]

$$\text{score}(X, S) = \log p(X \mid S) - \log p(X \mid \bar{S}). \tag{3}$$

This score is then compared with a predefined verification threshold. The speaker is accepted if the score exceeds the verification threshold, and otherwise rejected. We assume a common (global) threshold for all speakers.

The problem in the computation of (3) is that the likelihood of the alternative hypothesis is not directly accessible since this requires information of *all other speakers of the world*. There are two main approaches for the estimation of $p(X \mid \bar{S})$ [41]: *universal background model* (or *world model*) and *cohort set*. The world model is generated from a large set of speakers, and it attempts to model speech in general. In the cohort approach, for each client speaker, an individual set of cohort speakers is defined. Usually the cohort set contains the nearest speakers to the client, since intuitively these are the "best" impostors to the client speaker. We are not aware of large-scale comparison of the world model and cohort approaches, and it seems that currently there is no consensus which one of these is more accurate.

Cohort normalization methods can be divided into two classes: those that select the cohort speakers *offline* in the training phase [43], and those that select the cohort *online* [44] based on the closeness to the test vector sequence X. The online approach, also known as *unconstrained cohort normalization* (UCN) [41], [44], has been observed to be more accurate [42], [44], probably due to its adaptive nature. Another desirable feature of the UCN is that it does not require updating of the cohort sets when new speakers are enrolled in the system.

The usefulness of the online cohort selection is limited by its computational complexity. The computation of the normalized score (3) includes searching the cohort speakers, whose time increases linearly with the number of cohort candidates. Ariyaeeinia and Sivakumaran [44] noticed that a smaller equal error rate (EER) is obtained, if the cohort is selected among the client speakers instead of using an external cohort set.

TABLE I
SUMMARY OF THE CORPORA USED

	TIMIT	NIST
Language	English	English
Speakers	630	230
Speech type	Read speech	Conversational
Quality	Clean (hi-fi)	Telephone
Sampling rate	8.0 kHz	8.0 kHz
Quantization	16-bit linear	8-bit μ-law
Training speech (avg.)	21.9 sec.	119.0 sec.
Evaluation speech (avg.)	8.9 sec.	30.4 sec.

We propose to use pre-quantization for reducing the computational load of cohort search (see Algorithm 6). The input sequence X is first quantized into a smaller set $\hat{X}$ using the LBG algorithm [31], and majority of the speakers are pruned out based on the scores $D_{\text{avg}}(\hat{X}, C_i), i = 1, \ldots, N$. The remaining set of $K > 1$ best scoring speakers constitutes the cohort for the client speaker. The client score is also computed using the quantized sequence, and the normalized match score is computed as the ratio between the client score and average cohort speaker score. A small value indicates that the client score deviates clearly from the impostor distribution. The control parameters of the algorithm are the cohort size (K) and the size of the quantized test set (M).

Algorithm 6 Fast Cohort Scoring (FCS)

Let X be the unknown speaker's feature vectors ;
Let C_S be the claimed speaker's codebook ;
Let $K > 1$ be the desired cohort size ;
$\hat{X} :=$ LBG-Clustering(X, M) ;
Let $Coh := K$ best scoring speakers based on $D_{avg}(\hat{X}, C_i)$, excluding the client ;
$score(X, S) = D_{avg}(\hat{X}, C_S) / \frac{1}{K} \sum_{i \in Coh} D_{avg}(\hat{X}, C_i)$;

In acoustically mismatched conditions, both the client and cohort scores are expected to degrade, but their ratio is assumed to remains the same. This is the fundamental rationale behind score normalization. In other words, we assume

$$\frac{D_{\text{avg}}(X, C_S)}{\sum_j D_{\text{avg}}(X, C_j)} \approx \frac{D_{\text{avg}}(\hat{X}, C_S)}{\sum_k D_{\text{avg}}(\hat{X}, C_k)} \tag{4}$$

where j and k go over the indices of the cohort speakers selected using X and $\hat{X}$, respectively. The approximation (4) is good when X and $\hat{X}$ follow the same probability distribution.

V. EXPERIMENTS

A. Speech Material

For the experiments, we used two corpora, the *TIMIT* corpus [39] and the *NIST 1999 speaker recognition evaluation corpus* [45]. The TIMIT corpus was used for tuning the parameters of the algorithms, and the results were then validated using the NIST corpus.

Main features of the evaluated corpora are summarized in Table I. For consistency, the TIMIT files were downsampled from 16 to 8 kHz. This was preceded by alias cancellation using a digital low-pass FIR filter. TIMIT contains 10 files for each speaker, of which we selected 7 for training and 3 for testing. The files "sa" and "sx" having the same phonetic content for all speakers were included in the training material.

To our knowledge, no speaker identification experiments have been performed previously on the NIST-1999 corpus, and therefore, we needed to design the test setup ourselves. We selected to use the data from the male speakers only. Because we do not apply any channel compensation methods, we selected the training and recognition conditions to match closely. For training, we used both the "a" and "b" files for each speaker. For identification, we used the one speaker test segments from the same telephone line. In general it can be assumed that if two calls are from different lines, the handsets are different, and if they are from the same line, the handsets are the same [45]. In other words, the training and matching conditions have very likely the same handset type (electret/carbon button) for each speaker, but different speakers can have different handsets. The total number of test segments for this condition is 692.

B. Feature Extraction, Modeling, and Matching

We use the standard MFCCs as the features [29]. A pre-emphasiz filter $H(z) = 1 - 0.97z^{-1}$ is used before framing. Each frame is multiplied with a 30 ms Hamming window, shifted by 20 ms. From the windowed frame, FFT is computed, and the magnitude spectrum is filtered with a bank of 27 triangular filters spaced linearly on the mel-scale. The log-compressed filter outputs are converted into cepstral coefficients by DCT, and the 0th cepstral coefficient is ignored. Speaker models are generated by the LBG clustering algorithm [31]. The quantization distortion (1) with Euclidean distance is used as the matching function.

C. Performance Evaluation

The recognition accuracy of identification is measured by identification error rate, and the accuracy of the verification is measured by the equal error rate (EER). The methods were implemented using C/C++ languages. All experiments were carried out on a computing cluster of two Dell Optiplex G270 computers, each having 2.8 GHz processor and 1024 MB of memory. The operating system is Red Hat Linux release 9 with 2.4.22-openmosix2 kernel. We use system function `clock` divided by the constant `CLOCKS_PER_SEC` to measure the running time.

VI. RESULTS AND DISCUSSION

A. Baseline System

First, a few preliminary tests were carried out on the TIMIT corpus in order to find out suitable silence detection threshold. The number of MFCCs and model sizes were fixed to 12 and 64, respectively. With the best silence threshold (lowest error rate), about 11–12% of the frames were classified as silent and the average identification time improved by about 10% as compared without silence detection. Recognition accuracy also improved slightly when silence detection was used ($626/630$ correct $\rightarrow$ $627/630$ correct). Using the same silence detection threshold on the NIST, only 2.6% of the frames were classified as silent, and there was no improvement in the identification time.

TABLE II
PERFORMANCE OF THE BASELINE SYSTEM (TIMIT)

Codebook size	Error rate (%)	Avg. id. time (s)	
		Full search	VPT
8	10.5	0.29	0.33
16	2.22	0.57	0.62
32	0.63	1.15	1.11
64	0.48	2.37	2.07
128	0.16	4.82	4.14
256	0.16	10.2	8.21
512	0.32	21.6	12.9
No model	1.59	42.8	23.7

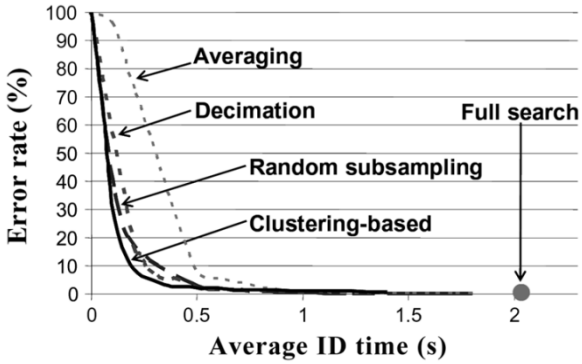


Fig. 7. Comparison of the PQ methods with codebook size 64 (TIMIT).

The effect of the number of MFCCs was studied next. Increasing the number of coefficients improved the identification accuracy up to 10–15 coefficients, after which the error rates stabilized. For the rest of the experiments, we fixed the number of coefficients to 12.

Table II summarizes the performance of the baseline system on the TIMIT corpus. The identification times are reported both for the full-search and for the VPT-indexed code vectors. The last row (no model) shows the results for using all training vectors directly as the speaker model as suggested in [46]. Increasing the model size improves the performance up to $M = 256$. After that, the results start to detoriate due to the overfitting effect, as observed also in [47]. The identification time increases with the codebook size. For small codebooks, VPT indexing does not have much effect on the identification times, but it becomes effective when $M \geq 32$. For the rest of the experiments, VPT indexing is used.

B. Pre-Quantization

Next, we compare the pre-quantization methods with codebook size fixed to $M = 64$. Parameters were optimized with extensive testing for each PQ method separately. The best time-error curves for each method are shown in Fig. 7. We observe that the clustering PQ gives the best results, especially at the low-end when time is critical. In general, PQ can be used to reduce the time about to 50% of the full search with a minor degradation in the accuracy.

C. Speaker Pruning

Next, we evaluate the performance of the speaker pruning variants with the pre-quantization turned off and speaker model size fixed to 64. Several experiments were carried out in order to find out the critical parameters. First, the variants were considered individually (see Figs. 8–11).

For the SP algorithm, we fixed the pruning interval ($M = 5, 10, 15$ vectors) and varied the number of pruned speakers (K). The shortest pruning interval ($M = 5$) gives the poorest results and the largest interval ($M = 15$) the best. The difference between $M = 10$ and $M = 15$ is relatively small.

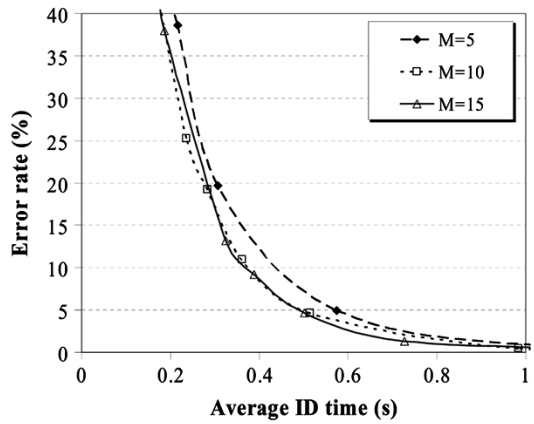


Fig. 8. Performance of the SP algorithm for different pruning intervals (TIMIT).

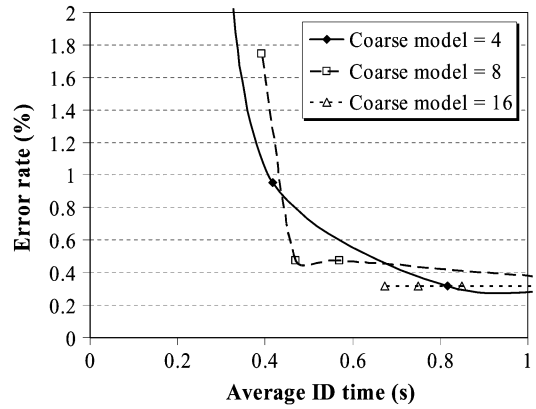


Fig. 9. Performance of the HP algorithm for different coarse model sizes with detail model size 64 (TIMIT).

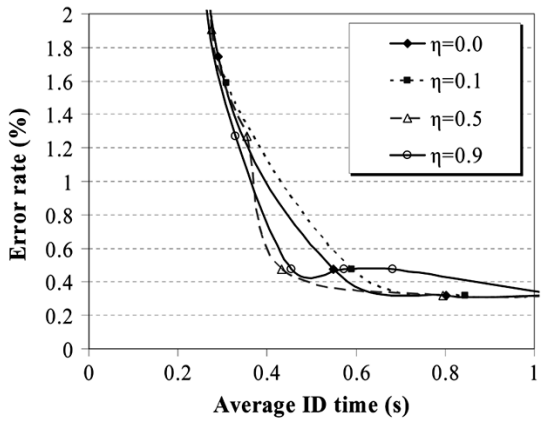


Fig. 10. Performance of the AP algorithm for different pruning thresholds (TIMIT).

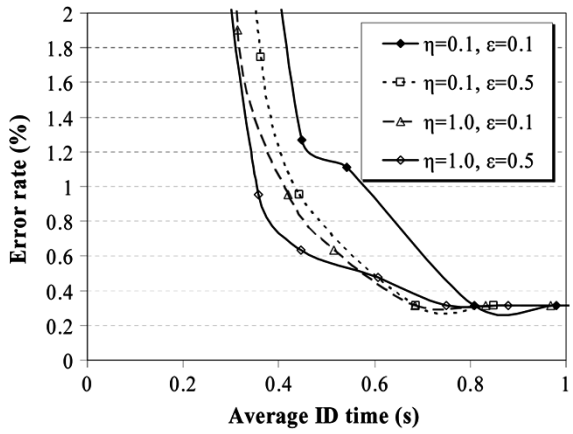


Fig. 11. Performance of the CP algorithm for different parameters (TIMIT).

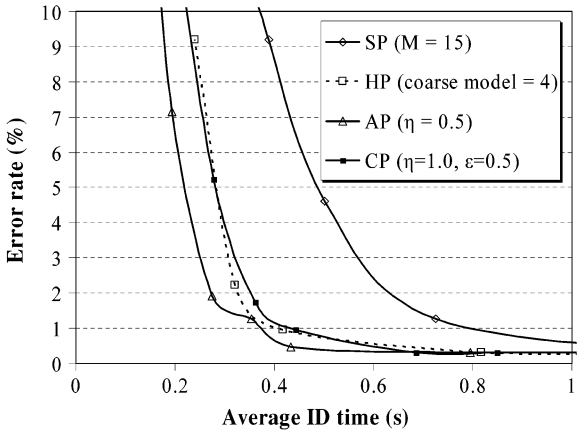


Fig. 12. Comparison of the pruning variants with speaker model size 64 (TIMIT).

For the HP algorithm, we fixed the coarse speaker model size $(M = 4, 8, 16)$ and varied the number of pruned speakers (K). We observe that the model sizes $M = 4$ and $M = 8$ give the best trade-off between the time and identification accuracy. If the codebook size is increased, more time is spent but the relative gain in accuracy is small.

For the AP algorithm, we fixed the parameter η in (2) to $\eta = \{0.0, 0.1, 0.5, 0.9\}$ and varied the pruning interval (M). The values $\eta = 0.5$ and $\eta = 0.9$ give the best results.

For the CP algorithm, we fixed the two thresholds $(\epsilon = 0.1, 0.5; \eta = 0.1, 1.0)$ and varied the pruning interval. The best result is obtained with combination $\eta = 1.0, \epsilon = 0.5$. The selection of the stabilization threshold ϵ seems to be less crucial than the pruning parameter η.

The pruning variants are compared in Fig. 12. The AP variant gives the best results, whereas the static pruning gives the poorest results. Next, we select the best PQ and pruning variants as well as the combination of PQ and pruning (PQP) as described in Section 3.5 and compare their performance. From the Fig. 13 we observe that the pruning approach gives slightly better results. However, in a time-critical application PQ might be slightly better. The combination of pre-quantization and pruning (PQP) gives the best result as expected.

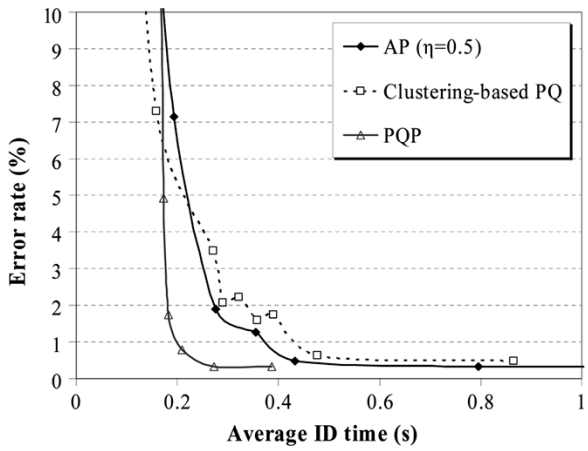


Fig. 13. Comparison of the best PQ and speaker pruning variants with speaker model size 64 (TIMIT).

TABLE III
SUMMARY OF THE BEST RESULTS ON THE TIMIT CORPUS

Setup	Vector quantization (VQ)				Gaussian mixture model (GMM)			
	Model size	Error rate (%)	Time (s)	Speed-up factor	Model size	Error rate (%)	Time (s)	Speed-up factor
Baseline	64	0.32	2.07	1:1	8	0.95	0.93	1:1
PQ		0.64	0.48	4:1		0.95	0.49	2:1
Pruning		0.48	0.43	5:1		1.11	0.21	4:1
PQP		0.32	0.27	8:1		0.95	0.21	4:1
Baseline	128	0.00	4.14	1:1	16	0.16	1.77	1:1
PQ		0.64	0.59	7:1		0.48	0.77	2:1
Pruning		0.00	1.88	2:1		0.16	0.92	2:1
PQP		0.00	0.31	13:1		0.16	0.18	10:1
Baseline	256	0.00	8.21	1:1	32	0.32	3.47	1:1
PQ		0.64	1.18	7:1		0.32	0.72	5:1
Pruning		0.00	3.28	3:1		0.32	1.80	2:1
PQP		0.00	0.65	13:1		0.32	0.40	9:1

D. Validation With NIST and GMM

Since TIMIT is known to give overly optimistic performance due to its laboratory quality and lack of intersession data, we validate the results on the NIST corpus. The best pre-quantization and pruning variants are also generalized to GMM modeling [14] as follows. Instead of using the log-likelihood $\log p(X \mid \mathrm{GMM}_i)$ as score, we use $-\log p(X \mid \mathrm{GMM}_i)$ instead. In this way, the scores are interpreted as dissimilarities, and the algorithms do not require any changes. We used diagonal covariance GMMs since they are widely used with the MFCC features, and they require significantly less computation and storage.

The best results for both corpora and model types are summarized in Tables III and IV. For pre-quantization, we use the clustering-based method, and for the pruning we use the adaptive variant. For the combination, we selected the clustering PQ and static pruning.

We optimized the model sizes for VQ and GMM separately. For VQ, larger codebook give more accurate results on both corpora as expected. GMM, on the other hand, is more sensitive to the selection of the model size. With TIMIT, model sizes larger than 64 degraded results dramatically (for model size 256 the error rate was 16.5%). There is simply not enough training data for robust parameter estimation of the models. For NIST, there is five times more training data, and therefore large models can be used.

TABLE IV
SUMMARY OF THE BEST RESULTS ON THE NIST 1999 CORPUS

Setup	Vector quantization (VQ)				Gaussian mixture model (GMM)			
	Model size	Error rate (%)	Time (s)	Speed-up factor	Model size	Error rate (%)	Time (s)	Speed-up factor
Baseline	64	18.06	2.92	1:1	64	17.34	9.58	1:1
PQ		18.20	0.62	5:1		18.79	0.73	13:1
Pruning		19.22	0.48	6:1		19.36	0.82	12:1
PQP		18.06	0.50	6:1		17.34	0.94	10:1
Baseline	128	17.78	5.80	1:1	128	17.05	18.90	1:1
PQ		18.93	0.64	9:1		18.20	0.84	23:1
Pruning		18.49	0.86	7:1		17.34	2.88	7:1
PQP		17.78	0.67	9:1		17.63	1.34	14:1
Baseline	256	17.34	11.40	1:1	256	16.90	37.93	1:1
PQ		18.20	0.70	16:1		18.50	1.11	34:1
Pruning		17.49	1.46	8:1		17.48	5.78	7:1
PQP		17.49	0.90	13:1		18.06	2.34	16:1

The problem of limited training data for GMM parameter estimation could be attacked by using, instead of the maximum likelihood (ML) training, the maximum a posteriori parameter (MAP) adaptation from the world model as described in [15]. Taking advantage of the relationship between the world model and the speaker-depended GMMs, it would also possible to reduce the matching time [15], [20]. In this paper, however, we restricted the study on the baseline ML method.

From the results of Tables III and IV we can make the following observations.

- Identification time depends on the size and the type of the model.
- The error rates are approximately of the same order for both VQ and GMM. For TIMIT, the error rates are close to zero, and for NIST they are around 17–19%.
- The speed-up factor of PQ increases with the model size as expected. Relative speed-up is higher for GMM than for VQ. Improvement of the pruning, on the other hand, depends much less on the model size.
- With TIMIT, PQP doubles the speed-up relative to PQ. With NIST, on the other hand, the PQP is not successful.
- The best speed-up factor for NIST with VQ is 16:1 increasing the error rate from 17.34% to 18.20%. For GMM, the corresponding speed-up factor is 34:1 with the increase of the error rate from 16.90% to 18.50%.

In general, we conclude that the results obtained with TIMIT hold also for NIST although there are differences between the corpora. More importantly, the studied algorithms generalize to GMM-based modeling. In fact, the speed-up factors are better for GMM than for VQ on the NIST corpus. The optimized systems are close to each other both in time and accuracy, and we cannot state that one of the models would be better than the other in terms of time/error trade-off. The ease of implementation, however, makes the VQ approach more attractive. In fact, prototype implementation for Symbian series 60 operating system for mobile devices is currently in progress.

The combination of PQ and GMM gives a good time-accuracy trade-off, which is consistent with the verification experiments carried out by McLaughlin *et al.* [18]. They noticed that the test sequence could be decimated up to factor 20:1 with minor effect on the verification performance. They found out that the fixed decimation (every Kth vector) gave the best performance. However, as we can see from the Fig. 7, the clustering based pre-quantization performs better. This explains partially why we obtained a better speed-up (up to 34:1).

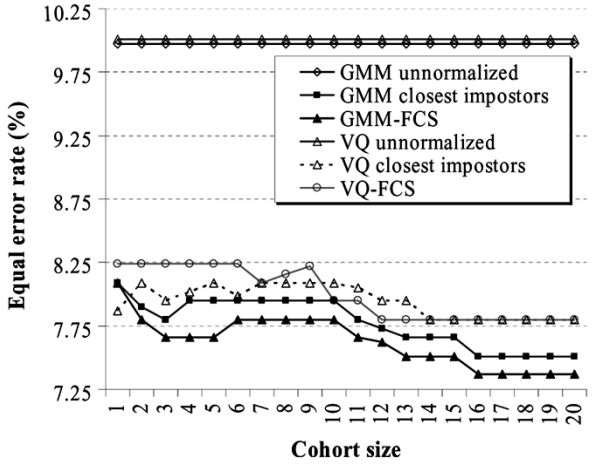


Fig. 14. Effect of the cohort size using different scoring methods ($\text{model sizes} = 128; M = 32$) (NIST).

E. Fast Cohort Scoring for Verification

The proposed cohort normalization method (FCS) was studied next on the NIST corpus. We used the same subset for verification than for the identification experiments, thus consisting of $N = 692$ genuine speaker trials and $N(N-1)/2 = 239\,086$ impostor trials. The speaker model size was set to 128 for both VQ and GMM based on the identification results, and the PQ codebook size for the FCS method was set to 32 after preliminary experiments. In both normalization methods, the client score is divided by the average cohort score. In the case of VQ, models are scored using the quantization distortion, and in the case of GMM, the log likelihood.

We consider the following methods:

- no normalization;
- closest impostors to the test sequence;
- fast cohort scoring (FCS).

The cohort size is varied from $K = 1$ to $K = 20$. The equal error rates of the normalization methods are shown in Fig. 14, along with the unnormalized case as a reference. We observe an decreasing trend in EER with increasing cohort size for both normalization methods and for both modeling techniques. GMM gives better results for both normalization methods. More interestingly, the proposed method (FCS) outperforms the method of closest impostors even though only the quantized test sequence is used for scoring. This result supports the claim that redundancy in the test sequence should be removed. The result also indicates that the assumption (4) holds in practice.

Table V summarizes the performances of the two score normalization methods. The speed-up factor is relative to the closest impostors method. The proposed method speeds up the verification by a factor of 23:1 and it also decreases the error rate at the same time. The equal error rates are relatively high in general, which is because of a simple acoustic front-end. We did not apply either delta processing nor channel compensation methods such as cepstral mean subtraction.

TABLE V
SUMMARY OF THE COHORT SELECTION METHODS
(cohort size = 20; model sizes = 128; $M = 32$) (NIST)

Method	Model	EER (%)	Avg. verif. time (s)	Speed-up factor
Closest impostors	VQ	7.80	5.79	1:1
	GMM	7.51	18.94	1:1
FCS	VQ	7.48	0.65	9:1
	GMM	6.94	0.84	23:1

VII. CONCLUSION

A real-time speaker identification system based on vector quantization (VQ) has been proposed. The most dominating factors of the identification time are the number of test vectors and the number of speakers. We used silence detection and pre-quantization for the reduction of the vectors, and speaker pruning for the reduction of the speakers. A VPT tree was applied for speeding up the nearest neighbor search from the speaker codebook.

We used the TIMIT corpus for tuning the parameters, and validated the results using the NIST-1999 speaker recognition evaluation corpus. With TIMIT, a speed-up factor of 13:1 was achieved without degradation in the identification accuracy. With NIST, a speed-up factor of 16:1 was achieved with a small degradation in the accuracy (17.34% versus 18.20%).

We demonstrated that the methods formulated for VQ modeling generalize to GMM modeling. With TIMIT, a speed-up factor of 10:1 was achieved. With NIST, a speed-up factor of 34:1 was achieved with a small degradation (16.90% versus 18.50%) in the accuracy.

We also applied pre-quantization for efficient cohort normalization in speaker verification. The proposed method turned out to be both faster and more accurate than the commonly used method of closest impostors. An EER of 6.94% was reached in average verification time of 0.84 s when the length of test utterance is 30.4 s, with a speed-up of 23:1 compared to the widely used closest impostors method.

Regarding the selection between pre-quantization and pruning methods, the former seems more attractive in the light of the experimental results on the NIST corpus, and the findings reported in [18]. Clustering can be effectively applied for removing redundancy from the test sequence with small or no degradation in the accuracy. A possible future direction could be toward developing more adaptive pre-quantization methods (all pre-quantization methods studied here assume either fixed buffer or codebook size).

In this paper we restricted the study of the GMM to the baseline ML method. However, it is expected that the studied methods generalize to the UBM/GMM framework [15] and further speedups are possible by combining UBM/GMM with pre-quantization and speaker pruning. It is also possible to use UBM idea in the VQ context in the same way by generating a large speaker-independent codebook and adapting the speaker-dependent codebooks from it.

Finally, it must be noted that the acoustic front-end was fixed to MFCC processing in this study, and it seems that further speed optimization with these features is difficult. A possible future direction could be to use multiparametric classification: a rough estimate of the speaker class could be based on pitch features, and the matching could then be refined using spectral features. Alternatively, one could use initially high-dimensional features, such as a combination of cepstrum, delta-parameters, F0 features and voicing information, followed by a mapping into a low-dimensional space by linear discriminant analysis (LDA), principal component analysis (PCA), or neural networks. In this way, probably more discriminative low-dimensional features could be derived.

REFERENCES

[1] S. Furui, "Recent advances in speaker recognition," *Pattern Recognit. Lett.*, vol. 18, no. 9, pp. 859–872, 1997.

[2] J. Campbell, "Speaker recognition: A tutorial," *Proc. IEEE*, vol. 85, no. 9, pp. 1437–1462, Sep. 1997.

[3] D. A. Reynolds, "An overview of automatic speaker recognition technology," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2002)*, Orlando, FL, 2002, pp. 4072–4075.

[4] A. Martin and M. Przybocki, "Speaker recognition in a multi-speaker environment," in *Proc. 7th Eur. Conf. Speech Communication and Technology (Eurospeech 2001)*, Aalborg, Denmark, 2001, pp. 787–790.

[5] I. Lapidot, H. Guterman, and A. Cohen, "Unsupervised speaker recognition based on competition between self-organizing maps," *IEEE Trans. Neural Networks*, vol. 13, pp. 877–887, 2002.

[6] D. Liu and F. Kubala, "Fast speaker change detection for broadcast news transcription and indexing," in *Proc. 6th European Conf. Speech Communication and Technology (Eurospeech 1999)*, Budapest, Hungary, 1999, pp. 1031–1034.

[7] S. Kwon and S. Narayanan, "Speaker change detection using a new weighted distance measure," in *Proc. Int. Conf. on Spoken Language Processing (ICSLP 2002)*, Denver, CO, 2002, pp. 2537–2540.

[8] R. Kuhn, J.-C. Junqua, P. Nguyen, and N. Niedzielski, "Rapid speaker adaptation in eigenvoice space," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 6, pp. 695–707, Nov. 2000.

[9] X. He and Y. Zhao, "Fast model selection based speaker adaptation for nonnative speech," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 4, pp. 298–307, Jul. 2003.

[10] W. Jia and W.-Y. Chan, "An experimental assessment of personal speech coding," *Speech Commun.*, vol. 30, no. 1, pp. 1–8, 2000.

[11] A. Glaeser and F. Bimbot, "Steps toward the integration of speaker recognition in real-world telecom applications," in *Proc. Int. Conf. on Spoken Language Processing (ICSLP 1998)*, Sydney, NSW, Australia, 1998.

[12] H. S. M. Beigi, S. H. Maes, J. S. Sorensen, and U. V. Chaudhari, "A hierarchical approach to large-scale speaker recognition," in *Proc. 6th Eur. Conf. Speech Communication and Technology (Eurospeech 1999)*, Budapest, Hungary, 1999, pp. 2203–2206.

[13] S. Furui, *Digital Speech Processing, Synthesis, and Recognition*, 2nd ed. New York: Marcel Dekker, 2001.

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

[15] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Process.*, vol. 10, no. 1, pp. 19–41, 2000.

[16] B. Sun, W. Liu, and Q. Zhong, "Hierarchical speaker identification using speaker clustering," in *Proc. Int. Conf. Natural Language Processing and Knowledge Engineering 2003*, Beijing, China, 2003, pp. 299–304.

[17] R. Auckenthaler and J. S. Mason, "Gaussian selection applied to text-independent speaker verification," in *Proc. Speaker Odyssey: The Speaker Recognition Workshop (Odyssey 2001)*, Crete, Greece, 2001, pp. 83–88.

[18] J. McLaughlin, D. A. Reynolds, and T. Gleason, "A study of computation speed-ups of the GMM-UBM speaker recognition system," in *Proc. 6th European Conf. Speech Communication and Technology (Eurospeech 1999)*, Budapest, Hungary, 1999, pp. 1215–1218.

[19] B. L. Pellom and J. H. L. Hansen, "An efficient scoring algorithm for gaussian mixture model based speaker identification," *IEEE Signal Process. Lett.*, vol. 5, no. 11, pp. 281–284, Nov. 1998.

[20] B. Xiang and T. Berger, "Efficient text-independent speaker verification with structural gaussian mixture models and neural network," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 5, pp. 447–456, Sep. 2003.

[21] M. Liu, E. Chang, and B. Q. Dai, "Hierarchical gaussian mixture model for speaker verification," in *Proc. Int. Conf. on Spoken Language Processing (ICSLP 2002)*, Denver, CO, 2002, pp. 1353–1356.
[22] Z. Pan, K. Kotani, and T. Ohmi, "An on-line hierarchical method of speaker identification for large population," in *Proc. NORSIG 2000*, Kolmården, Sweden, 2000.
[23] T. Kinnunen, E. Karpov, and P. Fränti, "A speaker pruning algorithm for real-time speaker identification," in *Proc. Audio- and Video-Based Biometric Authentication (AVBPA 2003)*, Guildford, U.K., 2003, pp. 639–646.
[24] J. Uhlmann, "Satisfying general proximity/similarity queries with metric trees," *Inform. Process. Lett.*, vol. 40, pp. 175–230, 1991.
[25] F. K. Soong, A. E. Rosenberg, B.-H. Juang, and L. R. Rabiner, "A vector quantization approach to speaker recognition," *AT&T Tech. J.*, vol. 66, pp. 14–26, 1987.
[26] J. He, L. Liu, and G. Palm, "A discriminative training algorithm for VQ-based speaker identification," *IEEE Trans. Speech Audio Process.*, vol. 7, no. 3, pp. 353–356, May 1999.
[27] T. Kinnunen and I. Kärkkäinen, "Class-discriminative weighted distortion measure for VQ-based speaker identification," in *Proc. Joint IAPR International Workshop on Statistical Pattern Recognition (S+SPR2002)*, Windsor, ON, Canada, 2002, pp. 681–688.
[28] G. Singh, A. Panda, S. Bhattacharyya, and T. Srikanthan, "Vector quantization techniques for GMM based speaker verification," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 2003)*, Hong Kong, 2003.
[29] J. R. Deller Jr., J. H. L. Hansen, and J. G. Proakis, *Discrete-Time Processing of Speech Signals*, 2nd ed. New York: IEEE Press, 2000.
[30] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Englewood Cliffs, NJ: Prentice-Hall, 2001.
[31] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. 28, no. 1, pp. 84–95, 1980.
[32] T. Kinnunen, T. Kilpeläinen, and P. Fränti, "Comparison of clustering algorithms in speaker identification," in *Proc. IASTED Int. Conf. Signal Processing and Communications (SPC 2000)*, Marbella, Spain, 2000, pp. 222–227.
[33] R.-H. Wang, L.-S. He, and H. Fujisaki, "A weighted distance measure based on the fine structure of feature space: application to speaker recognition," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 1990)*, Albuquerque, NM, 1990, pp. 273–276.
[34] T. Matsui and S. Furui, "A text-independent speaker recognition method robust against utterance variations," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 1991)*, Toronto, ON, Canada, 1991, pp. 377–380.
[35] A. L. Higgins, L. G. Bahler, and J. E. Porter, "Voice identification using nearest-neighbor distance measure," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP 1993)*, Minneapolis, MN, 1993, pp. 375–378.
[36] T. Kinnunen and P. Fränti, "Speaker discriminative weighting method for VQ-based speaker identification," in *Proc. Audio- and Video-Based Biometric Authentication (AVBPA 2001)*, Halmstad, Sweden, 2001, pp. 150–156.
[37] N. Fan and J. Rosca, "Enhanced VQ-based algorithms for speech independent speaker identification," in *Proc. Audio- and Video-Based Biometric Authentication (AVBPA 2003)*, Guildford, U.K., 2003, pp. 470–477.
[38] E. Chavez, G. Navarro, R. Baeza-Yates, and J. L. Marroquin, "Searching in metric spaces," *ACM Comput. Surv.*, vol. 33, no. 3, pp. 273–321, 2001.
[39] Linguistic Data Consortium (2004, Sept.). [Online]. Available: http://www.ldc.upenn.edu/
[40] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed. London, U.K.: Academic, 1990.
[41] R. Auckenthaler, M. Carey, and H. Lloyd-Thomas, "Score normalization for text-independent speaker verification systems," *Digital Signal Process.*, vol. 10, pp. 42–54, 2000.
[42] Y. Zigel and A. Cohen, "On cohort selection for speaker verification," in *Proc. 8th Eur. Conf. Speech Communication and Technology (Eurospeech 2003)*, Geneva, Switzerland, 2003, pp. 2977–2980.
[43] R. A. Finan, A. T. Sapeluk, and R. I. Damper, "Impostor cohort selection for score normalization in speaker verification," *Pattern Recognit. Lett.*, vol. 18, pp. 881–888, 1997.
[44] A. M. Ariyaeeinia and P. Sivakumaran, "Analysis and comparison of score normalization methods for text dependent speaker verification," in *Proc. 5th Eur. Conf. Speech Communication and Technology (Eurospeech 1997)*, Rhodes, Greece, 1997, pp. 1379–1382.
[45] A. Martin and M. Przybocki, "The NIST 1999 speaker recognition evaluation—An overview," *Digital Signal Process.*, vol. 10, pp. 1–18, 2000.
[46] D. R. Dersch and R. W. King, "Speaker models designed from complete data sets: A new approach to text-independent speaker verification," in *Proc. 5th Eur. Conf. Speech Communication and Technology (Eurospeech 1997)*, Rhodos, Greece, 1997, pp. 2323–2326.
[47] R. Stapert and J. S. Mason, "Speaker recognition and the acoustic speech space," in *Proc. Speaker Odyssey: The Speaker Recognition Workshop (Odyssey 2001)*, Crete, Greece, 2001, pp. 195–199.

Tomi Kinnunen received the M.Sc., Ph.Lic., and Ph.D. degrees in computer science from the University of Joensuu, Finland, in 1999, 2004, and 2005, respectively

Currently he is with the Institute for infocomm Research (IR2), Singapore. His research topics include automatic speaker recognition and speech signal processing.

Evgeny Karpov received the M.Sc. in applied mathematics from Saint Petersburg State University, Russia, in 2001, and the M.Sc. degree in computer science from the University of Joensuu, Finland, in 2003, where he is currently pursuing the Ph.D. degree.

Since 2005, he has been with the Multimedia Technologies Laboratory, Nokia Research Center, Tampere, Finland. His research topics include automatic speaker recognition and signal processing algorithms for mobile devices.

Pasi Fränti received the M.Sc. and Ph.D. degrees in computer science from the University of Turku, Finland, in 1991 and 1994, respectively.

From 1996 to 1999, he was a Postdoctoral Researcher (funded by the Academy of Finland) with the University of Joensuu, Finland, where he has been a Professor since 2000. His primary research interests are in image compression, pattern recognition, and data mining.