Fast Speech Processing Algorithms for Real-Time Identification of Wanted Persons using Digital Communication Networks

By

Muhammad Afzal
2004-PhD-CS-07

Supervised by

Professor Dr. Mohammad Ali Maud

A thesis submitted for partial fulfillment of the degree of Doctor of Philosophy in Computer Science to the University of Engineering and Technology, Lahore, Pakistan.

Department of Computer Science and Engineering,
Faculty of Electrical Engineering,
University of Engineering and Technology,
Lahore, Pakistan

2012

Certificate of Approval

It is certified that the research work presented in this thesis entitled "*Fast Speech Processing Algorithms for Real-Time Identification of Wanted Persons using Digital Communication Networks*" was conducted by
Mr. Muhammad Afzal under the supervision of Professor Dr. Mohammad Ali Maud.
The contribution of work is original and worthy for computer scientists

_____ Supervisor/ Internal Examiner Professor Dr. Mohammad Ali Maud The Department of Computer Science and Engineering, University of Engineering and Technology, Lahore, Pakistan	_____ Local Reviewer/ External Examiner Dr. Mian Muhammad. Awais, Associate Professor, Computer Science Lahore University of Management Sciences (LUMS), DHA, Lahore Cantt., Pakistan
_____ Professor Dr. Mohammad Ali Maud Chairman The Department of Computer Science and Engineering, University of Engineering and Technology, Lahore, Pakistan	_____ Professor Dr. Zubair A. Khan Dean The Faculty of Electrical Engineering, University of Engineering and Technology, Lahore, Pakistan

This thesis was evaluated by the following examiners

Internal Examiner/Research Supervisor

1. Professor Dr. Mohammad Ali Maud,
   Department of Computer Science and Engineering
   University of Engineering and Technology,
   Lahore, Pakistan
   Email: mamaud@uet.edu.pk

External Examiner from Pakistan

1. Dr. Mian Muhammad. Awais,
   Associate Professor, Computer Science,
   School of Science and Engineering
   Lahore University of Management Sciences (LUMS)
   DHA, Lahore Cantt., Pakistan
   Email: awais@lums.edu.pk

External Examiners from Abroad

1. Dr. Gul Nawaz Khan,
   Associate Professor and Program Director for Computer Engineering
   Department of Electrical and Computer Engineering
   Ryerson University, 350 Victoria Street, Toronto,
   Ontario M5B 2K3, Canada.
   Email: gnkhan@ee.ryerson.ca

2. Dr. Dil Mohammad Akbar Hussain
   Associate Professor,
   Department of Energy Technology,
   Aalborg University, Niels Bohrs Vej 8,
   6700 Esbjerg, Denmark.
   Email: akh@et.aau.dk

3. Dr. Mohammad Usman
   Fellow Scientist and Vice President of Engineering,
   Masimo, Incorporated,
   Mission Viejo, CA 92691
   Email: usman92691@gmail.com

Dedicated to my parents and my wife

Acknowledgement

Alhamdulillah, all praise is due to Allah who blessed me with all the abilities that permitted me to accomplish this arduous research work. The work was conducted in the Department of Computer Science and Engineering, University of Engineering and Technology, Lahore, Pakistan. Support of university for providing me education, employment and residence can not be over gratefully acknowledged.

The title of my research work was finalized by Professor Dr. Shaiq A. Haq, ex-chairman of the Department of Computer Science and Engineering. He could not continue supervising my research project and left the department at the on set of my work. Mainly, I received supervision from Professor Dr. Mohammad Ali Maud, current chairman of Department of Computer Science and Engineering. The knowledge and synergy that I used to accomplish this difficult research task definitely flowed through him. His correcting critique always guided me proximally on to the work track destined to the completion of my PhD work. His cooperation to facilitate a working environment in the department can not be over emphasized.

All my teachers who imparted knowledge to educate me at any level deserve my due respect and appreciation. Thank you all, may Allah bless you. Guiding and supporting critique of Dr. Ali Hammad Akbar and Dr. Khalid Mehmood Aamir has gone a along way in helping me to conduct my work, from time to time. I have been well regarding frequent comments and push of my colleagues to redefine my stops and goes during my arduous research work.

I am thankful to Linguistic Data Consortium (LDC) Pennsylvania University, USA for providing standard speaker recognition data for experimentation.

Life is about ones family. I owe due appreciation to the suffering of my wife and children from my neglects because of long professional and research work hours that kept me away from home. I thank my family for the prayers for me.

Abstract

Telephony networks are frequently connected to computers for speech processing to extract useful information such as automatic speaker identification (ASI). Matching of feature vectors extracted from speech sample of an unknown speaker, with models of registered speakers is the most time consuming component in real-time speaker identification systems. Time controlling parameters are size d and count T of extracted test feature vectors as well as size M, complexity and count N of models of registered speakers. Reported speedup techniques for Vector quantization (VQ) and Gaussian mixture model (GMM) based ASI systems reduce test feature vector count T by pre-quantization and reduce candidate registered speakers N by pruning unlikely models which introduces accuracy degradation. Vantage point tree (VPT) indexing of code vectors has also been used to decrease the effect of parameter M on ASI speed for VQ based systems. Somehow parameter d has remained unexplored in ASI speedup studies.

Speedup techniques for VQ based and GMM based real-time ASI without loss of accuracy are presented in this thesis. Speeding up closest code vector search (CCS) is focused for VQ based systems. Capability of partial distortion elimination (PDE), through reducing d parameter of codebook, was found more promising than VPT to speedup CCS. Advancing in this direction, speech signal stationarity has been capitalized to a greater extent than previously proposed technique of cluster size based sorting of code vectors to speedup PDE. Proximity relationship among code vectors established through Linde Buzo Gray (LBG) process of codebook generation has been substantiated. Based upon the high correlation of proximate code vectors, circular partial distortion elimination (CPDE) and toggling-CPDE algorithms have been proposed to speedup CCS. Further speedup for ASI is proposed through test feature vector

sequence pruning (VSP) when a codebook proves unlikely during search of best match speaker. Empirical results presented in this thesis show that an average speedup factor up to 5.8 for 630 registered speakers of TIMIT 8kHz corpus and 6.6 for 230 speakers of NIST-1999 database have been achieved through integrating VSP and TCPDE.

Speeding up potential of hierarchical speaker pruning (HSP) for faster ASI has also been demonstrated in this thesis. HSP prunes unlikely candidate speakers based on ranking results of coarse speaker models. Best match is then found from the detailed models of remaining speakers. VQ based and GMM based ASI systems are explored in depth for parameters governing the speedup performance of HSP. Using the smallest possible coarse model and pruning the largest number of detailed candidate models is the key objective for speedup through HSP. City block distance (CBD) is proposed instead of Euclidean distance (EUD) for ranking speakers in VQ based systems. This allows use of smaller codebook for ranking and pruning greater number of speakers. HSP has been ignored by previous authors for GMM based ASI systems due to discouraging speedup results in their studies of VQ-based systems. However, we achieved speedup factors up to 6.61 and 10.40 for GMM based ASI systems using HSP for 230 speaker from NIST-1999 and 630 speakers from TIMIT data, respectively. While speedup factors of up to 22.46 and 34.78 are achieved on TIMIT and NIST-1999 data for VQ based systems, respectively. All the speedup factors reported are with out any accuracy loss.

List of Acronyms and Symbols

ASI	Automatic speaker identification
C	Codebook – set of M code vectors or mean vectors
CBD	City block distance
CCB	Coarse codebook
CCS	Closest code vector search
CGMM	Coarse GMM
C_s	Codebook of a speaker with index s
CPDE	Circular partial distortion elimination
CSLU	Center of spoken language understanding of Oregon Health and Science University (OHSU), Portland, Oregon, USA
CSPDE	Cluster size based partial distortion elimination
c	Code vector of a detailed codebook
c'	Code vector of a coarse codebook
d	Size of MFCC feature vector or mean vector
D	Distortion or dissimilarity measure between test vectors and models of registered speakers
DCB	Detailed codebook
D_{eu}	Dissimilarity measure between test vectors and models of registered speakers based on Euclidean distance
DGMM	Detailed GMM
D_{mahal}	Dissimilarity measure between test vectors in X and models of registered speakers based on Mahalanobis distance
D'	Threshold minimum total distortion
DCT	Discrete cosine transform
ΔLPCC	First order difference of LPCC
ΔMFCC	First order difference of MFCC
EUD	Euclidean distance
$e(x_i, C)$	Distortion between i-th test feature vector x_i and a codebook C

$\mathbb{F} \in \mathbb{R}^{N \times M}$ Repository of weight factors of all registered GMMs

FFT Fast Fourier transform

f_{Lin} Linear frequency

f_{Mel} Mel frequency

GMM Gaussian mixture model

H Entropy of code vectors in a codebook

$H/\log_2 M$ Normalized entropy of code vectors in a codebook

λ_s GMM of speaker with index s

LBG Linde Buzo Gray

LPCC Linear predictive cepstral coefficient

M Codebook size -- number of c in a codebook

M' Size of coarse codebook -- number of c' in a C'

MFCC Mel frequency cepstral coefficient

$\mathfrak{M} \in \mathbb{R}^{N \times M \times d}$ Repositories of mean vectors of all registered GMMs

μ_m m-th mean vector of GMM

N Number of registered speakers in an ASI system

NIST National Institute of Standards and Technology, USA

PCM Pulse coded modulation

PDE Partial distortion elimination

PDEs Refers to any one of PDE, CSPDE, CPDE or TCPDE algorithm or all of them

pdf Probability density function

P_m Mixture weight for m-th state or mixture of a GMM

$\mathbb{R}$ Real number space

σ' Threshold CBD or squared EUD between x and c

$\mathbb{S}^{-1} \in \mathbb{R}^{N \times M \times d \times d}$ Repository of inverse covariance matrices $\ddot{\Sigma}_m^{-1}$ for all registered GMMs

$|\mathbb{S}| \in \mathbb{R}^{N \times M}$ Repository of determinant values of covariance matrices $\left|\ddot{\Sigma}_m\right|$ for all registered GMMs

Σ Summation notation

$\ddot{\Sigma}_m$	Covariance matrix of $\tilde{X}$ for μ_m
$\left\|\ddot{\Sigma}_m\right\|$	Determinant of covariance matrix of $\ddot{\Sigma}_m$
$\ddot{\Sigma}_m^{-1}$	Inverse of covariance matrix $\ddot{\Sigma}_m$
T	Number of x vectors in X
$\tilde{T}$	Number of $\tilde{x}$ vectors in $\tilde{X}$
TCPDE	Toggling circular partial distortion elimination
TIMIT	TI (Texas Instrument) and MIT (Massachusetts Institute of Technology)
UBM	Universal background model
VPT	Vantage point tree
VQ	Vector quantization
VSP	Vector sequence pruning
vi	Code vector index found closest to the previous test vector
W	Number of speaker models pruned out in HSP
X	Vector sequence extracted from test speech sample for speaker testing
$\tilde{X}$	Vector sequence extracted from training speech sample for speaker model training
x_i	An i-th test vector of X
$\tilde{x}_{\tilde{i}}$	An $\tilde{i}$-th training vector of $\tilde{X}$

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

Use of digital speech processing for speaker identification is growing day by day. Campbell [1] defines automatic speaker recognition (ASR) as an activity of a machine to recognize an individual from a spoken sentence or a phrase. ASR has to work in two modes namely learning mode and recognition mode. The system prepares speaker template or model in the learning mode. Learning mode is also termed as training mode. In the recognition mode, the system compares the speech sample of an unknown speaker with models developed in the training mode.

The term automatic speaker recognition is also used to express two types of machine activities, namely automatic speaker verification (ASV) and automatic speaker identification (ASI). An ASV system evaluates a speaker's claim to be a person known to it [1] [2]. This claim is either accepted or rejected based on a system learned threshold value of similarity measure compared with value of the similarity measure computed from current speaker's voice sample. ASI system tries to find the best match to the features of test speech of a speaker from learned feature models of speakers stored in the speaker database.

Along with linguistic contents, speech also carries personal information of the speaker like age, gender, health, mood, intention etc. Speech signal transmitted through a telephone network is degraded due to channel and mouth piece microphone. ASI somehow utilizes speaker specific features that are intricately ingrained in the speech signal. The field of ASI falls under the larger umbrella of biometric person identification that uses physical or learned traits of a person [3] [4].

1.1 Motivation

Digital surveillance for crime controlling through telephone networks demands further research to speedup ASI. Modern day crimes are planned and communicated through telephone networks. Habitual criminals involved in such activities can be trapped through network sniffing and voice matching. Such ASI systems must facilitate easy and quick addition and deletion of voice models from the database of wanted persons or registered speaker. Fast speech processing algorithms implemented for voice pattern matching can detect appearance of wanted persons as they use digital telephone network. Apart from services like voice dialing, telephone banking, shopping over telephone network, information and reservation, ASI is also used as a forensic tool.

ASI have been used in [5] for verification tasks in telecommunication applications. Lower bit rate speaker specific codec has been demonstrated in [6]. Improved user specific services are delivered by ASI front end by adapting speech processing systems. Real time speaker identification requirement underpin need of speeding up feature matching techniques.

A number of techniques, for speeding up ASV, have been proposed for systems based on Gaussian mixture model (GMM), considered as the state of the art technique [7] [8] [9]. However, these speedup techniques do not allow easy removal and addition of speaker models from the database of registered speakers. Kinnunen et al. [10] presents study on ASV and ASI optimization for speedup based on vector quantization (VQ). They also applied speedup techniques proving best for VQ systems on to their GMM systems. However, their reported speedup factors come with accuracy degradation.

1.2 Scope of Research Work and Methodology

This thesis proposes speedup techniques for VQ and GMM based ASI systems without accuracy loss. ASI systems based on GMM/Universal Background Model (UBM) are not considered because such systems require re-training of UBM and GMMs of registered speakers when a new speaker is added to the database or some registered speaker is removed from it. This drawback is crucial for ASI systems that frequently update the database of registered speakers. This difference is discussed in related work. Speedup performance of previous works is also compared with results of our ASI systems.

Most of the time spent in ASI is consumed during matching models of the registered speaker with features of the test sample [10]. We focused on accelerating pattern matching unit of ASI, so that our proposed techniques can be implemented into the pattern matching modules of existing ASI systems to increase their efficiency regardless of feature set used in the systems. However, Mel-frequency cepstral coefficients (MFCC) were used in this thesis to evaluate speedup efficiency of proposed techniques.

Literature on ASI in general and specifically on ASI speedup was reviewed to identify grey areas to investigate those aspects. VQ and GMM based ASI systems were built to test the performance of novelty introduced using C# programming language of Microsoft .NET framework. C# programming language compares well with C/C++ programming language for execution efficiency. Speech databases used for experimentation were CSLU [11], TIMIT [12] and NIST-1999 [13]. TIMIT corpus consists of clean microphone speech stored in linear pulse coded modulation (PCM) format while CSLU and NIST-1999 corpora contain recorded telephone conversation stored in linear and μ–law PCM encoding, respectively.

Speech files were selected from the corpora in such a way that text independent ASI experiments can be conducted for all the registered speakers. Feature vectors of training samples were extracted to train VQ and GMM speaker models which were subsequently stored. Feature vectors of test samples were also computed once and stored to evaluate the performance of ASI systems. Speedup factors were computed with reference to baseline full search system for time spent in feature vector matching like McLaughlin et al., [9] and Kinnunen et al., [10]. Average speaker identification time was calculated from a batch run time for testing all registered speakers by noting start and end times accurate to millisecond with the help of 'System.DateTime' method of C# .NET.

1.3 Contribution

This thesis has identified a number of aspects that contribute to speedup of ASI, such as, capability of partial distortion elimination (PDE) for speeding up VQ-based ASI systems is better than a number of previously proposed techniques. New insight to codebook structure imparted by LBG algorithm is highlighted in terms of utilization of multiple scan orders through VQ codebook that efficiently utilizes stationarity of speech signal to speedup ASI. GMM is parametric enhancement of VQ modeling and classification paradigm. Hierarchical speaker pruning (HSP) was found to be insufficiently explored in previous studies. HSP speedups ASI by quickly pruning unlikely speaker from list of candidate speakers by comparing smaller coarse speaker models and then matching detailed models of fewer likely ones. Use of CBD is proposed for coarse codebooks instead of EUD to prune larger number of candidate speakers for faster ASI through HSP. HSP has also been demonstrated to be very efficient for GMM-based ASI systems.

The rest of the thesis includes background of ASR systems in chapter 2. Chapter 3 presents review of related on speedup ASI systems and discusses drawbacks of previous techniques. Chapter 4 proposes new speedup techniques for efficient ASI systems. Chapter 5 describes experiments conducted to evaluate the proposed techniques. Chapter 5 also presents the empirical results and discussions. Conclusions are made in chapter 6 while chapter 7 recommends future work needed.

Chapter 2
Automatic Speaker Recognition Systems

Voice is the most natural biometric for person identification. A telephony facility suffices to identify a person remotely. It can be done tacitly and elusively for non volunteering subjects without notice while he or she is making a call. Keys and passwords can be forgotten while authenticating or can be stolen easily. Speech serves for secure authentication and verification.

Forensic cases requiring identification of crime perpetrators that are heard and not seen because of masking or darkness can be done through speaker identification techniques. Long speech recording done through wiretapping can be searched by applying ASI to track a given speaker for forensic investigation. Organized crimes are mostly planned by previous criminals and communicated through telephone networks. ASI can play a vital role in trapping the criminals at large and pre-empting such acts. Recently applications of ASI methodology towards decision support in forensics have been more emphasized [14] [15] [16] [17] [18]. However, in forensic application of ASI techniques output a measure of similarity for the human operator to decide about its significance. Whereas commercial ASI systems make hard decision about the identity of a naturally communicating speaker and do not require special attention from her or him.

Voice biometrics can be added to other automatic authentication methods like face recognition, iris recognition and finger print recognition to enhance the authentication accuracy. Acoustic features used in speaker identification are also usable in speech recognition which allows advantageous combination of the tasks [19]. Speaker identification accuracy of computer systems and human has been compared in [14] [20]. Schmidt-Nielsen and Crystal compared speaker identification performance of 63 human listeners with ASI systems [20]. For clean

speech samples and matching acoustic conditions ASI systems outperformed human listeners while for noisy and unmatched acoustic conditions human listener were better than machines [14] [20]. Reportedly, the human listeners used pronunciation, accent, timbre, intonations and speaking rate to distinguish among speakers instead of spectral features as used by ASI systems [21]. However, human listeners learn to recognize voices by the passage of time and tend to become fuzzy as number persons to recognize from their speech increases. Therefore, experiments for comparison of human and modern machine, with latest algorithms running on them, need to be repeated again with larger set of registered of speakers.

2.1 Classification of Speaker Recognition Systems

Campbell [1] classifies ASR systems based on the following criteria:

- **Based on Speaker Membership**: Two categories are 'close set speaker recognition systems' and 'open set speaker recognition systems'. The former category of systems assumes that the test speaker would always be from the learned ones by the ASR system. The latter systems do not have such presupposition. They first find the best match and then compare the similarity measure with a threshold value for the identified speaker to 'accept' or 'reject' the test speaker.

- **Based on Test Input Speech Text**: There are mainly three categories on this basis namely 'text dependent speaker recognition systems', 'prompted text speaker recognition systems' and 'text independent speaker recognition systems'. Text dependent speaker recognition systems require some fixed text to be spoken in recognition time. Prompted text speaker recognition systems need the speaker to speak the system prompted text at recognition time. Text independent speaker recognition systems do not require any fixed text to be spoken.

2.2 Components of Speaker Recognition System

A general structure of a speaker recognition system is shown in Figure 2.1. The feature extraction component for both ASI and ASV systems is necessarily the same. It is used both during training and testing phases of an ARS system. This component transforms raw speech signal into a set of feature vectors having a meaningful and manageable structure. This transformation reduces statistically redundant data and enhances or at least retains speaker specific information. These feature vectors are further converted to an efficient structure called speaker model during training phase [22]. For spectral features GMM and VQ models are frequently considered as the baseline models [7] [10] [23].

The feature set extracted from the speech sample of a speaker during recognition mode is compared with stored models. The speaker matching component does comparisons to generate a real number measure of similarity or dissimilarity with each stored models in case of speaker identification or only with model of claimed identity in the case of speaker verification.

The logical working of decision component in case of identification is finding the best match id for close set. It accepts the best match or rejects the best match for open set identification. In case of verification the output is 'reject' or 'accept'. An other type of response can be to prompt like 'unable to decide', 'speak again', 'continue speaking' or 'please wait and speak again when the noise subsides'. The decision component may also respond as 'Unable to decide' for lower SNR.

2.3 Feature Selection Criteria

Speech is a complex signal [1] carrying vast amount of information, for example, the language of speech, text of the speech, identity of speaker, mood of the speaker, intention of the speaker, gender and age of the speaker and many other. To identify a speaker, enhancing features

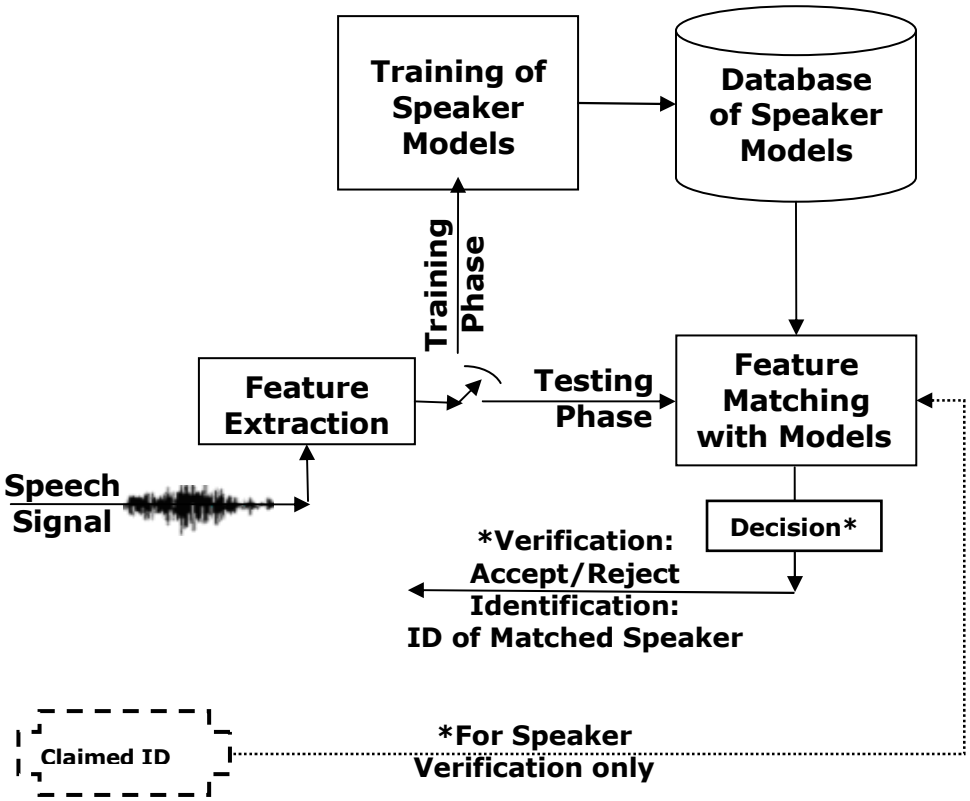

Figure 2.1: A general structure of components of automatic speaker recognition system

that contain identity information pertaining more to his or her physiological traits and suppressing others is desirable. Criteria for feature selection presented in [24] [25] are highlighted in the following.

- **Robustness:** The selected feature must least prone to noise, quantization and distortion.
- **Occurrence:** The selected feature must occur frequently in the speech so that even a small speech sample contains this feature.
- **Natural:** The selected feature must exist in speech naturally without any intention so that mood and intention do not alter it.
- **Impersonate:** The selected feature must be difficult to mimic to reduce false accepts.
- **Variability:** To increase the discriminative power of the classifier, the selected feature must have large inter speaker variation and small within speaker variation.

2.4 Types of Features

Numerous features have been proposed for speaker identification to encode speaker identity [26] [27][28][29][30][31][32][33]. They are categorized into following classes:

- **Time Domain:** for example, zero crossing rate, total energy and pauses in speech.
- **Spectral:** for example, Mel-frequency cepstrum co-efficient (MFCC), linear predictive cepstrum co-efficient (LPCC), line spectral frequency (LSF), long term average spectrum (LTAS), formant frequencies.
- **Dynamic:** for example, first order difference e.g., (ΔLPCC), (ΔMFCC), modulation frequencies.
- **Source:** for example, fundamental frequency F0, glottal pulse estimated shape.

- **Supra segmental:** for example, fundamental frequency contour, intensity contour, micro prosody.
- **High Level:** Idiosyncrasy styles, like pronouncing /r in different style by Lahoreans.
- **Supervectors:** A number of small-dimensional vectors are cascaded to form high-dimensional vector called supervectors [34]. Any variation in different utterances of a speaker can be characterized by their supervectors whether it results from difference in networks, environments or phonetic contents [21]. However, supervectors are mostly used for ASV.

2.5 Reduction of Feature Vector Size

Use of subset of features for efficient model preparation is performed by two methodologies namely feature selection and feature mapping.

2.5.1 Feature Selection Methodology

In feature selection, more effective component features are selected and less effective features are ignored. Dependency of classification performance due to feature selection on small samples has been analyzed by Jain and Zongker [35]. This methodology was introduced to speaker recognition domain by Sambur in [28] [36] and, Cheung and Eisenstein in [37]. They used dynamic programming (DP) for selection of efficient subset of features. Class separation maximizing capability of DP has been presented by Campbell in [1]. The feature selection task requires a criterion for component selection which may be optimized through a search strategy. Selection and trial of every component is not feasible. Better methods presented in [35] like genetic algorithm, top down and bottom up searches and dynamic programming are practical search strategies for component selection. Analytical comparison with reference to speaker recognition is given by Charlet and Jouvet [38].

2.5.2 Feature Mapping Methodology

In feature mapping different subsets of component features are analyzed to find their linear or non-linear combinations mapping function. This mapping function then replaces the selected subset of component features. Different component analysis techniques to find a mapping function for a subset of feature components are described briefly in the following:

- **LDA:** Linear Discriminant analysis follows the directions of maximizing linear separation between labeled classes.
- **ICA:** Independent component analysis is effective in finding linear mixing of sources.
- **PCA:** Principal component analysis tracks the largest variances and hence used to eliminate correlation between the features.

More details about the above analysis techniques are given in [39] [40].

2.6 Speaker Recognition Paradigms – Modeling and Matching

Selected speaker's specific feature set is extracted from the speech utterance of a speaker. Variation in the feature components, for example pitch, is randomly distributed over whole utterance. Modeling of the sequence of extracted feature sets can be done by taking average values on the whole utterance [3]. On the other hand dividing speech signal into small time frames (50-100 frames/sec) captures a semi-instantaneous value of the features [10][41]. Extraction of features from speech frames results into a sequence of feature vectors. Define $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \ldots, \tilde{x}_{\tilde{T}})$ as the sequence of $\tilde{T}$ feature vectors of a speech sample extracted during system training time, and define $X = (x_1, x_2, x_3, \ldots, x_T)$ as the sequence of T feature vectors of a person extracted during system testing time. For speaker identification, the two sequences are

compared through computing some dissimilarity measure. Hence speaker recognition requires sequence matching as defined by Equation (2.1).

$$D(X, \tilde{X}) \in \mathbb{R} \quad | \quad X \in \mathbb{R}^{T \times d}, \tilde{X} \in \mathbb{R}^{\tilde{T} \times d} \tag{2.1}$$

Where d is the number of elements of each feature vector extracted. Equation (2.1) computes a match score, based on a difference criteria such as the Euclidean distance. The process of reducing count of vectors of $\tilde{X}$ to make an efficient speaker's representation is termed as speaker modeling. This is done to speedup computation of match score given by Equation (2.1). Campbell [1] defines following two categories of speaker models namely template models and stochastic models.

2.6.1 Template Models

2.6.1.1 No Model

Stored sequence $\tilde{X}$, after feature extraction during training, can be a simple template and matched with sequence X at testing time [42]. This scheme is called 'No Model' [10]. It can have maximum amount of information but suffers from excessive matching time. That is why number of training vectors is reduced in the template by clustering techniques into $M \,|\, M < T$ means of vectors [53].

2.6.1.2 Dynamic Time Warping (DTW)

This technique is suitable for text dependent speaker recognition due to its speaking rate compensation capability [44]. Template model for text dependent recognition is a sequence of templates $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \ldots, \tilde{x}_{\tilde{T}})$ which is developed during training and matched to input sequence $X = (x_1, x_2, x_3, \ldots, x_T)$ at testing time. Inconsistency inherent in the human speech results in unequal count of training feature vectors $\tilde{T}$ and count of test feature vectors T in

general. The following Equation (2.2) gives an asymmetric match score between the template and input sequence in terms of a dissimilarity measure.

$$D = \sum_{i=1}^{T} d(x_i, \tilde{x}_{j(i)}) \tag{2.2}$$

Here $d(x_i, \tilde{x}_{j(i)})$ is distance measure between i-th test vector and the matching j-th reference vector. The distance measure is absolute difference for single dimension vectors and Euclidean distance for multidimensional vectors.

2.6.1.3 Average Set Models

A model having an average vector $\tilde{\mu}$ of $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \ldots, \tilde{x}_{\tilde{T}})$ maintains a single template. It is compared with testing time average vector μ of $X = (x_1, x_2, x_3, \ldots, x_T)$ by minimum distance classifier [2] using Euclidean distance as dissimilarity measure between $\tilde{\mu}$ and μ as given by Equation (2.3).

$$D = \frac{1}{d} \sum_{i=1}^{d} (\mu[i] - \tilde{\mu}[i])^2 \tag{2.3}$$

Here $\mu[i]$ stands for i-th average component of the extracted feature vectors. In speaker verification task, the speaker is detected when D is less than some threshold value. In speaker identification task the speaker is chosen, from the target speakers, for which D is minimum. Euclidean distance dissimilarity measure between $\tilde{\mu}$ and test feature vectors $X = (x_1, x_2, x_3, \ldots, x_T)$ can also be based on the following Equation (2.4) [1] with $I \in \mathbb{R}^{T \times T}$ being an identity matrix.

$$D_{eu}(x_i, \tilde{\mu}) = (x_i - \tilde{\mu})' \, I (x_i - \tilde{\mu}) \quad | \, x_i \in X \tag{2.4}$$

Instead of identity matrix, Mahalanobis distance [39] based dissimilarity measure uses covariance matrix $\ddot{\Sigma}$ as given by Equation (2.5) [1]. The four dotted cap-superscript is used in this thesis for keeping covariance matrix $\ddot{\Sigma}$ distinct from summation notation $\sum$.

$$D_{mahal}(x_i, \tilde{\mu}) = (x_i - \tilde{\mu})' \ddot{\Sigma}^{-1} (x_i - \tilde{\mu}) \quad | x_i \in X \tag{2.5}$$

2.6.1.4 Vector Quantization (VQ) Model

Average set model has a drawback that it does not undertake different acoustic speech classes resulting from speech events occurring in the speaker's vocal tract. Averaging tends to blur those events like generation of quasi periodic sound during vowel; or impulse sound during starting words with /b, /p; or fricative noise during /s, /sh and other acoustic classes. Comparing similar sound classes with similar classes is more reasonable. Classification of speech into unlabelled acoustic classes could be supervised as well as unsupervised. VQ is an unsupervised classification technique which uses Linde Buzo Gray (LBG) algorithm or some other clustering algorithm for finding M means or centroids $C = (c_1, c_2, c_3, \ldots, c_M)$ of non-overlapping clusters in the speech data [22] [43] [45]. Each of M means vector, also called code vector, represents an identified acoustic class of a speaker model. This set of M means is called speaker's codebook. VQ model represents acoustic classes of events happening in the vocal tract of the speaker but does not consider temporal sequence of the acoustic events.

VQ-based ASI systems employ average minimum quantization distortion classifier [1] [10]. This classifier compares each test feature vector of $X = (x_1, x_2, x_3, \ldots, x_T)$ with each centroid of $C = (c_1, c_2, c_3, \ldots, c_M)$ according to some distance measure and calculates a match score as average distortion through Equation (2.6) [10].

$$D(X,C) = \frac{1}{T}\sum_{i=1}^{T} \min_{1 \le m \le M} d(x_i, c_m) \tag{2.6}$$

For speaker verification task the constraint, $D(X, C_{ta.rget}) < Threshold$ must be satisfied to accept a speaker. For speaker identification from N registered speakers best match-score is computed as minimum distortion to identify a speaker s^* with Equation (2.7).

$$Decision\ s^* = \arg\min_{1 \le s \le N} D(X, C_s) \tag{2.7}$$

Kinnunen et al., [10] used 'no model' scheme for comparison with other VQ models and showed that this scheme slows down speaker matching step the most, as compared with other VQ speaker models of various codebook sizes.

2.6.1.5 Nearest Neighbor

Nearest neighbor approach to speaker recognition was introduced by Higgins et al. [46]. Nearest neighbor method combines the strengths of DTW and VQ [1]. Their method keeps training or reference frames $\tilde{X} = (\tilde{x}_1, \tilde{x}_2, \tilde{x}_3, \ldots, \tilde{x}_{\tilde{T}})$ with temporal information intact and does not compute *M* mean clusters. Using squared Euclidean distance between reference frames and test speech frames $X = (x_1, x_2, x_3, \ldots, x_T)$ they have proposed the following match scoring Equation (2.8) [46].

$$\begin{aligned} D(X,C) = \; & \frac{1}{T}\sum_{i=1}^{T} \min_{1 \le k \le \tilde{T}} d(x_i, \tilde{x}_k)^2 + \frac{1}{\tilde{T}}\sum_{i=1}^{\tilde{T}} \min_{1 \le k \le T} d(x_i, \tilde{x}_k)^2 \\ & - \frac{1}{T}\sum_{i=1}^{T} \min_{1 \le k \le T, k \ne i} d(x_i, x_k)^2 - \frac{1}{\tilde{T}}\sum_{i=1}^{\tilde{T}} \min_{1 \le k \le \tilde{T}, i \ne k} d(\tilde{x}_i, \tilde{x}_k)^2 \end{aligned} \tag{2.8}$$

They have shown that the first two sum terms are proportional to cross entropies and last two sum terms are proportional to self entropies. Campbell [1] demonstrated that this match scoring scheme was one of the most powerful methods.

2.6.2 Stochastic Models

Same phonation vary every next time due to dynamic behaviors of vocal tract reshaping and glottal air flow resulting from speech context, co-articulation along with anatomical and aerodynamic variation [2]. This non-deterministic nature of a speaker's voice variation is not effectively captured and represented by the template based models. A multidimensional Gaussian probability density function **(pdf)** is a method to represent speaker's voice variations [47]. The Gaussian **pdf** is state dependent. Each acoustic sound class represents a state that has different Gaussian **pdf**.

Consider VQ speaker model consisting of M centroids or mean vectors representing various acoustic classes or states of vocal tract of the speaker. The Gaussian **pdf** of a random feature vector $x \in \mathbb{R}^d$ for m-th state is given by Equation (2.9).

$$b_m(x) = (2\pi)^{\frac{-d}{2}} \, |\ddot{\Sigma}_m|^{\frac{-1}{2}} \exp\left\{-(x-\mu_m)' \, \ddot{\Sigma}_m^{-1}(x-\mu_m)/2\right\} \qquad (2.9)$$

Where $|\ddot{\Sigma}_m|$ and $\ddot{\Sigma}_m^{-1}$ are determinant and inverse of the covariance matrix $\ddot{\Sigma}_m$ for state m represented by mean vector μ_m. Here, symbol μ_m is used in place of c_m to switch smoothly from VQ nomenclature to GMM nomenclature. The probability of a feature vector x to belong to m-th state of GMM is given by $P_m \, b_m(x)$. Where $b_m(x)$ is the component mixture density; and P_m is the mixture weight or component prior probabilities satisfying $\sum_{m=1}^{M} P_m = 1$. Mathematically, GMM consisting of M states symbolized by λ_s for a speaker $\boldsymbol{s}$ is represented by M triples, namely $(P_m, \mu_m, \ddot{\Sigma}_m)$, computed from training feature vectors $\tilde{X}$, where μ_m , $\ddot{\Sigma}_m$ and P_m are mean, covariance and weight factor of m-th state, respectively.

Given $\tilde{X}$ of a speaker ***s***, expectation maximization (EM) algorithm [48] is used to maximize likelihood of $\tilde{X}$ for λ_s by fine tuning GMM parameters iteratively. There are two match scoring methods for GMM based speaker identification, namely maximum a posteriori probability (MAP) and maximum likelihood (ML).

2.6.2.1 Maximum A Posteriori Probability (MAP)

For ASI, test feature vectors $X = (x_1, x_2, x_3, \ldots, x_T)$ are matched with GMM models of N registered speakers namely $(\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_N)$. Maximum a posteriori probability (MAP) approach computes the probability of model of each speaker $s \,|\, 1 \le s \le N$ for each test feature vector $x_i \in X \,|\, 1 \le i \le T$ using the Bayes' rule for conditional probability as given by Equation (2.10).

$$p(\lambda_s \mid x_i) = p(x_i \mid \lambda_s) p(\lambda_s) / p(x_i) \tag{2.10}$$

Since $p(x_i)$ is constant for all speaker models λ_s, it suffices to compute $p(x_i \mid \lambda_s) p(\lambda_s)$ and maximize it for all test vectors over all the speaker models.

2.6.2.2 Maximum Likelihood Classification

In this case, likelihood $p(x_1, x_2, x_3, \ldots, x_T \,|\, \lambda_s)$ of each $x_i \in X$ to belong to a target speaker ***s*** is maximized as given by Equation (2.11).

$$p(X \,|\, \lambda_s) = p(x_1, x_2, x_3, \ldots, x_T \,|\, \lambda_s) = \prod_{i=1}^{T} p(x_i \,|\, \lambda_s) \tag{2.11}$$

Application of logarithm to the above Equation (2.11) represents the speaker identification solution through Equation (2.12) [2].

$$Decision \quad s^* = \arg\max_{1 \le s \le N} \sum_{i=1}^{T} \log[p(x_i \mid \lambda_s)] \tag{2.12}$$

2.6.2.2 Mono-Gaussian Model

It is a special case of GMM of a speaker that uses single Gaussian state defined by a pair $\left(\mu, \ddot{\Sigma}\right)$ that is estimated from training speech data [3]. Mono-Gaussian model is reported to be computationally efficient and satisfactory in results [1]. Besacier and Bonastre [3] reported that Mono-Gaussian was less accurate than GMM but computationally faster up to three times both in model training and speaker verification. The single mean vector may be ignored motivated from the fact that a single covariance matrix does not change by a constant bias. Bimbot et al., [49] experimented on TIMIT database and found that incorporating mean, when having laboratory quality clean speech during training and testing improved performance. While for degraded telephone speech NTIMIT and FTIMIT models based on covariance only, perform better [49].

2.6.3 Other Models

Neural networks (NN) have also been used for speaker identification. Younes Bennani [50] reported development of three types of speaker models namely NN, hidden Markov model (HHM) and multivariate auto-regressive model (M-ARM). He has also developed hybrid speaker models by combining NN with HMM and NN with M-ARM and has compared system performance on a subset of TIMIT database.

Trevor et al., in [51] used multi-layer perceptrons (MLP) for speaker identification based on 29-element vector speaker model. They implemented the system using probabilistic RAM (pRAM) neurons on VLSI circuit.

Vincet and Campbell [52] employed support vector machine (SVM) for speaker verification and speaker identification tasks using polynomial kernel. Polynomial classifier based on 2, 3 and 4 degree polynomials have been used for speaker recognition in [53] using LPCC and ΔLPCC feature vectors on YOHO speaker recognition database. This technique map low dimensional feature vectors to a high dimensional features. For example, 2-dimensional feature vector is mapped to 6-dimensional feature space by using a 2-degree polynomial model as given by mapping Equation (2.13).

$$[x_1 \;\; x_2] \Rightarrow [1 \;\; x_1 \;\; x_2 \;\; x_1^2 \;\; x_1 x_2 \;\; x_2^2] \tag{2.13}$$

Campbell et al., [53] report 2925-dimensional higher feature model, based on 3-degree polynomial, mapped from 24-dimensional vectors by concatenating 12 LPCC and 12 ΔLPCC which gives best speaker verification result on YOHO database. However, use of larger sized feature vectors enormously slow down the speed of ASI systems as will be discussed in the next chapter.

Chapter 3

Related Work on Speeding up ASI Systems

Real-time speaker identification emphasizes on reducing the time spent to identify a speaker. VQ based speaker identification as defined by Equation (2.6) and Equation (2.7) require comparison of $X \in \mathbb{R}^{d \times T}$ with each of N codebooks $C_s \in \mathbb{R}^{d \times M} \mid 1 \le s \le N$ in the database of registered speakers to find the best matching codebook. Most of the time is spent in conducting NT closest code vector searches (CCS) through computing of *d*-dimensional Euclidean distance (EUD) MNT times. Thus, VQ based ASI systems have a complexity order of $O(d\,M\,N\,T)$ [10].

Vantage point tree (VPT) used in [10] could speedup CCS by 24% for codebook of size 256. Although for the best case of balanced binary tree like indexing, VPT computes EUD of $O(\log_2 M)$ to complete a CCS in a codebook of size M [10]. The paper [10] also reports pre-quantization of feature vector to decrease T which may distort the speaker specific features embedded in X. Other speedup techniques in [10] use heuristic pruning of unlikely codebooks. On the whole, three parameters namely, M, N and T were manipulated in [10] by combining afore mentioned techniques to speedup ASI. The parameter *d* has been ignored in speedup by above mentioned researchers. Afzal and Haq [54] have considered the parameter d for speeding up ASI using PDE algorithm. They achieved substantial speedup on CSLU [11] and TIMIT [12] data. PDE, proposed in [55] and [56], has been previously utilized in codebook generation mainly for image processing.

PDE speedups CCS by terminating EUD distance computation when $\sum_{j=1}^{d} (x_i[j] - c_m[j])^2$ becomes greater or equal to latest minimum squared distance at $j < d$.

Previously proposed techniques of speeding up CCS are reclassified under 'approximation and elimination frame work' in [57]. Most of the techniques considered in [57] explicitly approximate the closest code vector to eliminate unlikely code vectors to mitigate the effect of parameter M . Explicit approximation computes a set of constraints by calculating delimiting values of axis through projection computation. During CCS elimination of code vectors is then achieved by enforcing those constraints on test vector elements. However, high overhead of approximation and elimination which increases with dimensionality is their major drawback [57]. An implicit scheme of approximation to enforce elimination through PDE that used reordering of code vectors in decreasing order of clusters size (CSPDE here after) was proposed in [58]. CSPDE in [58] used vectors of raw 8 kHz speech of 20 seconds length. Compared with plain PDE, the multiplication operations saved in CCS vary from 32.7% to 4.3% for codebooks sizes $\left\{2^d \times d \mid 4 \le d \le 10, d \in \mathbb{N}\right\}$, respectively [58]. High entropy of code vectors of larger vectors, which increased for larger codebooks, decreased the elimination capacity of CSPDE [57] [58]. It seems to perform low for ASI which requires calculation of $D(X,C)$ for all registered codebooks. That is, rearranged code vectors, of $N-1$ codebooks of dissimilar voices to that of X , might not speed up VQ-based ASI systems on the whole as elaborated in later chapters.

VQ codebook is non-parametric model while GMM is a parametric model. Training of GMM usually starts from VQ codebook of $\tilde{X}$ generated by LBG algorithm to compute other parameters of GMM like covariance matrices and weight factors for mean vector for soft clusters rather than hard cluster of VQ [2] [48]. EM algorithm is used to maximize log likelihood of $\tilde{X}$ for mean vectors of soft clusters of GMM. Generally, EM algorithm fine tunes GMM parameters $\left(\mu_m, \ddot{\Sigma}_m, P_m\right) \mid 1 \le m \le M$, in 5 to 7 iterations [2] [10]. To obviate redundant

evaluations of inverse matrices $\ddot{\Sigma}_m^{-1} \in \mathbb{R}^{d \times d}$ and determinants $\left|\ddot{\Sigma}_m\right| \in \mathbb{R}$ of covariance matrices $\ddot{\Sigma}_m$ during speaker identification mode, ASI systems instead store GMM of each registered speakers as M quintuples that is $(\mu_m, \ddot{\Sigma}_m^{-1}, \left|\ddot{\Sigma}_m\right|, P_m)$.

Let, for a GMM-based ASI system with N registered speakers, $\mathfrak{M} \in \mathbb{R}^{N \times M \times d}$, $\mathbb{S}^{-1} \in \mathbb{R}^{N \times M \times d \times d}$, $|\mathbb{S}| \in \mathbb{R}^{N \times M}$ and $\mathbb{F} \in \mathbb{R}^{N \times M}$ respectively represent stored repositories of mean vectors, inverse covariance matrices, determinants and weight factors of all trained GMMs. A GMM-based ASI system then identifies the test speaker of X by finding its maximum log-likelihood from N registered speaker models using Equation (3.1).

$$spkr\ \ id = \arg\max_{1 \le s \le N} \sum_{t=1}^{T} \log \sum_{m=1}^{M} \frac{\mathbb{F}_{s,m}}{\sqrt{(2\pi)^d \left|\mathbb{S}\right|_{s,m}}} \exp\left[-\frac{1}{2} (x_t - \mathfrak{M}_{s,m}) \mathbb{S}_{s,m}^{-1} (x_t - \mathfrak{M}_{s,m})' \right] \quad (3.1)$$

The matrices within the pair of square brackets in Equation (3.1), namely, $(x_t - \mathfrak{M}_{s,m})$, $(x_t - \mathfrak{M}_{s,m})'$ and $\mathbb{S}_{s,m}^{-1}$ respectively have order $1 \times d$, $d \times 1$ and $d \times d$. Hence, Equation (3.1) carries out multiplication, addition, square root, log and antilog for $NTM(4 + d + d^2)$, $NT(1 + M(1 + d + d^2))$, NTM, NT and NTM times, respectively. Therefore, the time order complexity of computing Equation (3.1) is given by $O(TNMd^2)$.

However, only diagonal of $\ddot{\Sigma}_m^{-1}$ matrix is used for comparable accuracy and much faster execution ASI systems as given by Equation (3.2) [10].

$$spkr\ \ id = \arg\max_{1 \le s \le N} \sum_{t=1}^{T} \log \sum_{m=1}^{M} \frac{\mathbb{F}_{s,m}}{\sqrt{(2\pi)^d \left|\mathbb{S}\right|_{s,m}}} \exp\left[\sum_{i=1}^{d} -\frac{1}{2} (x_{t,i} - \mathfrak{M}_{s,m,i})^2 \mathbb{S}_{s,m,i,i}^{-1} \right] \quad (3.2)$$

Equation (3.2) is faster and requires computing of multiplication and addition for $NTM(4+3d)$ and $NT(1+M(1+2d))$ number of times, respectively. The time order complexity of Equation (3.2) is given by $O(TNMd)$.

Randomly arranged registered N speakers' space requires $O(NTM)$ computations of **pdf** for full search speaker identification where M is the number of states of each speaker's GMM. ASI speeding up through hierarchical speaker pruning (HSP) [7] [8] [59] decreases the **pdf** computations to indirectly reduce the speed controlling parameters namely M and N. Hierarchical speaker pruning may be classified into two categories namely training time pruning and testing time pruning. Using adaptive technique Reynolds et al., [7] demonstrated that UBM allowed hierarchical structuring of the space of registered speakers during training time which reduced load of **pdf** computation. UBM is trained from speech samples of all speakers using EM algorithm. Bayesian adaptation of UBM is used to train GMM model of each registered speaker using training speech samples of individual speaker [7]. Each component of UBM can have an adapted GMM component for each speaker. Beigi et al., [59] demonstrated merging strategy for the development of training time hierarchical structure of speakers' space. In merging, individual GMM of all speakers are scanned and pairs of closely similar models are identified and merged to find a higher level model. The process is repeated until top level single GMM is obtained. Sun et al., [8] have reported identification time reduction to one sixth, with no accuracy loss for a database of 160 speakers using merging technique. They have implemented the ISODATA clustering algorithm and compared the results with those obtained from full search technique. GMM/UBM has a drawback that UBM is based on a very large number of GMM states which is then used to adapt to individual speaker specific GMM which increase the absolute identification time. More over, it is difficult to register new speakers or to remove registered speaker from the ASI systems that are based on training time hierarchical structures of speaker models.

HSP during testing time uses coarse speaker models to rank registered speakers and prunes out unlikely ones before final best match search with detailed models. Kinnunen et al., [10] also studied testing time HSP along with other speaker pruning techniques for VQ based models. Since their VQ-based HSP performed worse than adaptive pruning, they prematurely ignored HSP for GMM based study. However, HSP needs more exploration to improve its performance as elaborated in the next chapters.

3.1 Limitation of Existing Techniques

- Speeding up CCS with VPT is only up to 1.67 times of baseline full search for codebook sized 512 [10].
- CSPDE considered best to speedup PDE is known to reduce multiplications by a factor of 4.3% only for codebook sized 1024 [58].
- Training time HSP, that is, UBM/GMM requires very large number of states like 2048 for UBM and the speedup factor achieved is 2 or less [9]. More over UBM/GMM requires retraining of UBM and all GMMs when a speaker is removed from the system or added to it.
- Training time HSP with merging strategy used to build multi-level hierarchical structure of registered speakers' space has the same drawback as stated for UBM/GMM.
- Pre-quantization of X tends to distort the speaker specific information ingrained in X. The speedup factors achieved through pre-quantization as reported in [10] are with an accuracy loss.
- Adaptive testing time speaker pruning method speedups ASI by two times without accuracy loss [10].

Chapter 4

Efficient Techniques for ASI Systems

Although it is claimed in [58] that LBG-generated code vectors are not in a favorable order to speedup CCS of PDE. In fact, the codebook construction process of LBG algorithm imparts an intrinsic order to the code vectors. This order of code vectors can be utilized for efficient CCS. Departing from Voronoi view, refer [60], of VQ codebook, as shown in Figure 4.1(a), is required to get to the operative insight. Proximity relationship between the code vectors is highlighted in Figure 4.1(b) by showing how new code vectors are inserted in between by LBG algorithm.

LBG initially computes a codebook of one code vector and then progressively doubles the codebook size from the codebook generated in the last step. This is accomplished by splitting clusters of training vectors of previous codebook into a pair of smaller clusters and subsequently tuning the new clusters. Every cluster splitting of old cluster creates a new code vector away from the old code vector in one direction of the d axis and another new code vector correspondingly in opposite direction of the d axis in $\mathbb{R}^d$ space. The tuning step of LBG iteratively scans through the training vectors $\tilde{X}$ till VQ distortion of the current code vectors fails to improve. Each iteration checks membership of vectors of $\tilde{X}$ in the new clusters to update the code vector according to the latest membership and tests the improvement in distortion. A view of this progress during codebook development through LBG with defined relationship between indexes of code vectors in current and previous codebook is depicted in Figure 4.1(b). Therefore, a proximity trend exists among the code vectors of LBG-generated

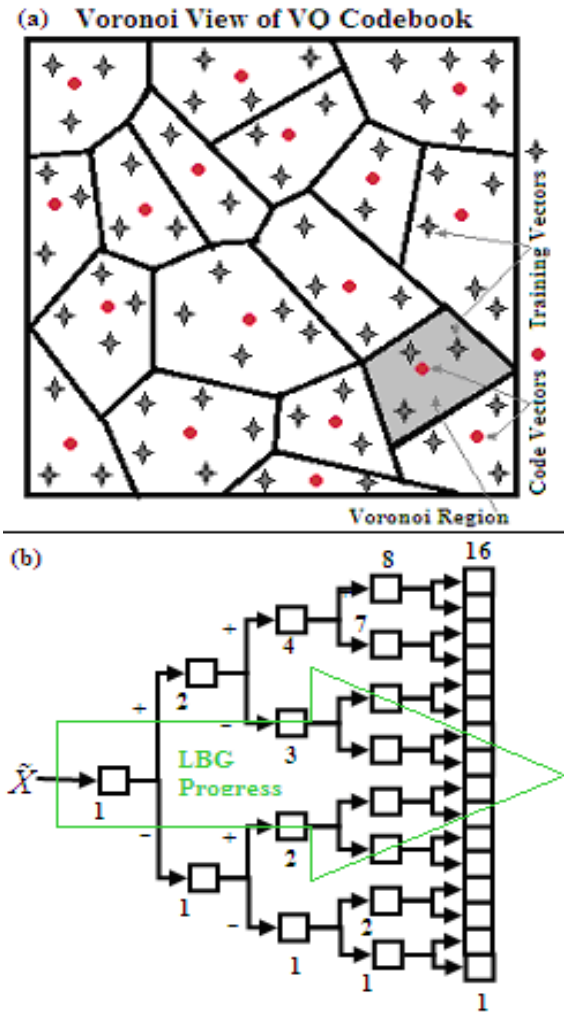

Figure 4.1: LBG codebook aspects: (a) Typical Voronoi view (b) LBG codebook generation view, more similar code vectors are assigned adjacent indexes and less similar are placed farther apart

codebook. The code vectors are placed adjacent in the codebook if they more similar and placed farther apart otherwise.

We propose CPDE and TCPDE algorithms that capitalize this proximity relationship of code vectors through forming multiple favorable scan routes to speedup CCS for computing distortion between a vector and a codebook symbolized as $e(x_i, C)$. These algorithms also efficiently utilize the stationarity in speech signal to achieve better approximation defined in [57]. For this purpose CPDE starts CCS from code vector indexed vi for each test vector. The value of vi must be set equal to the index of closest code vector found in the succeeding CCS. The value of vi may be set $1 \le vi \le M$ for x_1 by the calling Algorithm 3.

Algorithm 1: To compute vector distortion with CPDE

Inputs: M , d , vi , $x_i \in \mathbb{R}^d$, $C \in \mathbb{R}^{M \times d}$ **Output**: $e(x_i, C)$

1: Set $m = vi$; $\sigma' = \infty$; $k = 1$
2: Do
  2.1: Set $j = 1$; $\sigma = 0$
  2.2: Do
    2.2.1: Set $\sigma = \sigma + (x_i[j] - c_m[j])^2$
    2.2.2: *if* $\sigma \ge \sigma'$ *goto* 2.4:
    2.2.3: Set $j = j + 1$
  while $j \le d$
  2.3: Set $\sigma' = \sigma; vi = m$
  ▼ Select next code vector or select the first
  ▼ code vector if the last one was tested currently
  2.4: Set $m = m + 1;$ *if* $m > M$ *then* $m = 1$
  2.5: Set $k = k + 1;$
 while $k \le M$
3: Set $e(x_i, C) = \sqrt{\sigma'}$

CPDE reduces to PDE algorithm if step **2.4** and two assignments involving vi are eliminated, and k is replaced with m. CSPDE is essentially PDE with rearranged code vectors. CPDE is quicker than PDE to locate the closest code vector and to set σ' to the lower value

earlier. This results in elimination condition, $\sigma \geq \sigma'$, to occur at smaller values of j for the rest of the code vectors. CPDE conducts a unidirectional circular search through the codebook. CPDE suffers delay in CCS for the closest code vectors that lie in the opposite direction of its circular scan. To reduce this delay, CPDE is modified to check next adjacent code vectors by toggling between anti-clockwise and clockwise directions. Algorithm 2 (TCPDE) or toggling CPDE for faster CCS is given in the following.

Algorithm 2: To compute vector distortion using TCPDE

Inputs: M , d , vi , $x_i \in \mathbb{R}^d$, $C \in \mathbb{R}^{M \times d}$ **Output**: $e(x_i, C)$

1: Set $m = vi$; $m' = vi$; $\sigma' = \infty$; $k = 1$
2: Do
  2.1: Set $j = 1$; $\sigma = 0$
  ▼ Clockwise scan of codebook
  2.2: Do
    2.2.1: Set $\sigma = \sigma + (x_i[j] - c_m[j])^2$
    2.2.2: *if* $\sigma \geq \sigma'$ *goto* 2.4:
    2.2.3: Set $j = j + 1$
    while $j \leq d$
  2.3: Set $\sigma' = \sigma$; $vi = m$
  ▼ For clockwise scan select previous code vector
  ▼ or the last one if 1st was scanned currently
  2.4: Set $m = m - 1$; *if* $m < 1$ *then* $m = M$
  ▼ For anti-clockwise scan select previous code
  ▼ vector or the last one if 1st was scanned currently
  2.5: Set $m' = m' + 1$; *if* $m' > M$ *then* $m' = 1$
  2.6: Set $j = 1$; $\sigma = 0$
  ▼ Anti clockwise scan of codebook
  2.7: Do
    2.7.1: Set $\sigma = \sigma + (x_i[j] - c_{m'}[j])^2$
    2.7.2: *if* $\sigma \geq \sigma'$ *goto* 2.9:
    2.7.3: Set $j = j + 1$
    while $j \leq d$
  2.8: Set $\sigma' = \sigma$; $vi = m'$
  ▼ Advancing scan in both directions
  2.9: Set $k = k + 1$
 while $2 \times k \leq M$
3: Set $e(x_i, C) = \sqrt{\sigma'}$

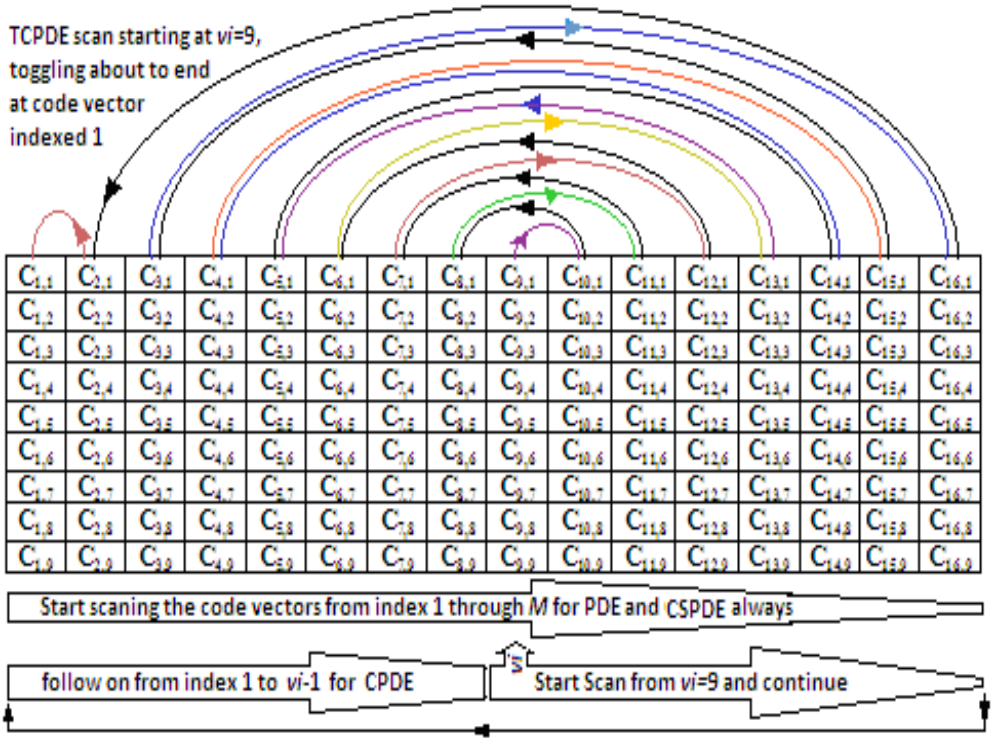

Figure 4.2: Typical search paths for PDE, CSPDE, CPDE and TCPDE for $vi = 9$ with codebook size M=16

TCPDE more frequently hits the closest code vector earlier than CPDE. Consequently higher level of elimination of TCPDE speedups CCS further. Typical search paths for various PDEs are shown in Figure 4.2.

Let the two consecutive steps '$\sigma = \sigma + (x_i[j] - c_{m'}[j])^2$' and '*if* $\sigma \geq \sigma'$ *goto label* :' be called **core-steps.** These steps are common in PDE, CSPDE, CPDE and TCPDE. The two steps perform (1, 2, 1) multiplications, additions and comparisons (MACs) of floating point values. A full CCS performs first core-step for $d \times M$ times and $\sigma \geq \sigma'$ comparison M times. Thus full CCS involves $M \times (d, 2d, 1)$ MACs. However, all variants of PDE actually perform the core-steps for $d' \times M$ times depending on their elimination performance. Therefore, they involve $M \times d' \times (1, 2, 1)$ MACs, where d' is average of j values at which $\sigma \geq \sigma'$ condition becomes true during a CCS. The worst, best and average case analysis is shown in Table 4.1 for all variants of PDE.

Both, CPDE and TCPDE perform M and $(1 + ½)M$ extra integer additions and integer comparisons than PDE, respectively. Floating point storage locations required to store codebook and test feature vectors are $d \times M \times N$ and $d \times T$, respectively. Single extra integer storage is required by CPDE and TCPDE for storing index of best matched code vector.

Instead of average distortion as used in [10], use of total distortion in Equation (2.6) opens a way of pruning X for unlikely codebooks without adding risk of accuracy degradation. Algorithm 3 uses this point to find the best matching codebook and thus conducts CCS less than $N \times T$ times. Respectively initializing si and D' by 1 and ∞, Algorithm 3 iteratively updates them each time $D(X, C_s) \leq D'$ condition becomes true.

Table 4.1: FLOPS analysis for all variants of PDE for best, worst and average case

Case	**Assuming $\boldsymbol{\sigma} \geq \boldsymbol{\sigma}'$ becomes true**	**MACs** $\times(1,2,1)$
Worst	at $j = d$ or never $\forall m \vert 1 \leq m \leq M$	$M \times d$
Best	at $j = 1$ $\forall m \vert 2 \leq m \leq M$	$M + d - 1$
Average	(worst + best) /2	$\frac{Md + M + d - 1}{2}$

Algorithm 3: To compute minimum speaker distortion using VSP

Inputs: d , M , N , T , $X \in \mathbb{R}^{T\times d}$, $C_s \in \mathbb{R}^{M\times d} \mid 1 \le s \le N$ **Output:** si

```
1: Initialize D' = ∞; s = 1; vi = 1;
2: Do
      2.1: Initialize i = 1 ; D = 0
      2.2: Do
             2.2.1: Set   D = D + e(x_i, C_s)
             2.2.2: if D ≥ D'  goto 2.4:
             2.2.3: Set    i = i + 1
          while  i ≤ T
      ▼ Update currently min distortion and best speaker Id
      2.3: Reset  D' = D; si = s
      2.4: Set  s = s + 1;
   while  s ≤ N
3: Output Test Speaker id = si
```

To compute $e(x_i, C_s)$ Algorithm 3 may use full CCS or a PDE variant. In this implementation, Algorithm 3 carries out CCS for $N \times T'$ times where T' is average of i values at which condition $D \ge D'$ occur. This way the algorithm results in vector sequence pruning (VSP) for codebooks that prove unlikely; because $T' < T$.

Kinnunen et al., [10] used coarse codebooks (CCB) of sizes 4, 8 and 16 along with detailed codebook (DCB) of size 64 in their HSP experimentation. They concluded from their results that testing time HSP is not efficient for VQ and completely ignored it for GMM based experiments. We realized that their HSP study was limited and hence explored it detailed. We tried to improve the ranking quality based on CCB.

An ASI system using coarse speaker models may rank the models based on their sorted order of quantization distortion or likelihood of X for registered codebooks or GMMs, respectively. Smaller rank of a registered speaker implies better ranking with rank of correctly identified speaker being the least that is 1st.

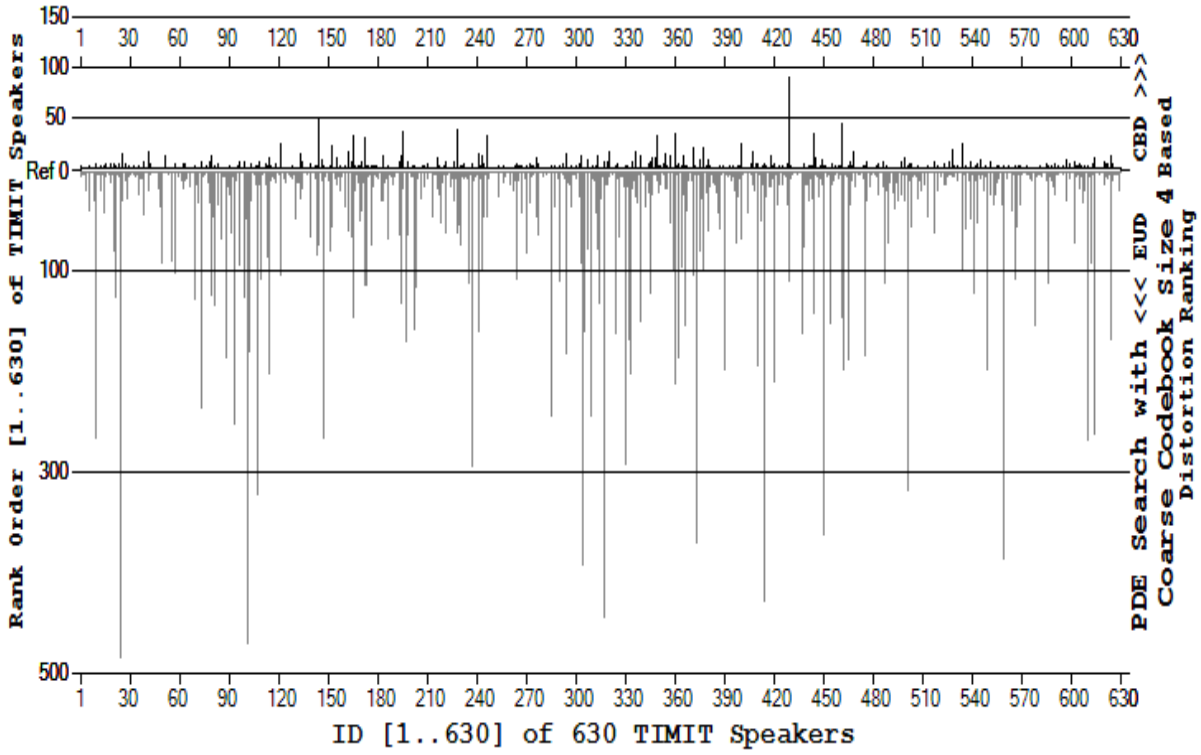

Figure 4.3: Ranking order comparison of CBD and EUD based ordering of VQ based systems for 630 TIMIT speakers

Ranking quality may be measured as variance or standard deviation of rank of all the registered speakers by some ASI system using coarse models. Ranking quality improves with size of speaker models. For VQ based systems distance measure also have profound effect on the ranking quality of ASI systems. Comparison of speaker ranking of EUD based and CBD based systems, as shown by Figure 4.3. A large room for improvement in HSP is evident in Figure 4.3. For HSP, the expressions M, C, $(c_m[j]-x_i[j])^2$ and $\sqrt{\sigma'}$ are replaced with M', C', $|c'_m[j]-x_i[j]|$ and σ', respectively, in PDE version of Algorithm 1 for coarse codebooks $C'=(c'_1,c'_2,c'_3,\ldots,c'_{M'})$ containing M' code vectors. We also observed that accuracy of VQ-based ASI systems for smaller size (<64) codebooks is higher with CBD than that with EUD. The computational time complexity of VQ-based ASI systems using CBD is also $O(TNMd)$. Therefore in HSP, CBD is proposed to rank speakers with Equation (4.1) using coarse codebooks. Worst ranked W speakers are pruned before best match searching based on EUD from the remaining DCBs.

$$D(X,C'_s)=\sum_{i=1}^{T}\min_{1\le m\le M}\sum_{j=1}^{d}|c'_m[j]-x_i[j]| \tag{4.1}$$

Improved HSP Algorithm 4 given in the following uses CCB for ranking with CBD based PDE and DCBs for identification test speaker with EUD based PDE by executing Algorithm 1 to compute $e(x_i,C_s)$.

For HSP studies of GMM based ASI systems, log-likelihood was computed according to Equation (4.2) using only diagonal of the covariance matrix inverse.

$$p(X|\lambda_s)=\sum_{t=1}^{T}\log\sum_{m=1}^{M}\frac{\mathbb{F}_{s,m}}{\sqrt{(2\pi)^d\,|\mathbb{S}|_{s,m}}}\exp\left(\sum_{i=1}^{d}-\frac{1}{2}(x_{t,i}-\mathfrak{M}_{s,m,i})^2\mathbb{S}^{-1}_{s,m,i,i}\right) \tag{4.2}$$

Algorithm 4: Improved HSP using CBD and EUD

Inputs: d , M , N , T , W , $C'_s \in \mathbb{R}^{M' \times d} \mid 1 \le s \le N$, $C_s \in \mathbb{R}^{M \times d} \mid 1 \le s \le N$

Output: *id*

1. Make a set S of all N registered speakers
2. $\forall s \in S$ Compute $D(X, C'_s)$ using CBD and sort
3. Prune out W worst speakers from S
4. $\forall s \in S$ Compute $D(X, C_s)$ using EUD
5. Output id of speaker's codebook with $\arg\min_{s \in S} D(X, C_s)$

For this purpose $D(X, C'_s)$ was replaced with $(-\log p(X \mid \lambda'_s))$ in HSP Algorithm 3. Step 2 used coarse GMM (CGMM) symbolized as λ'_s, for pruning W worst speakers whereas $M' = \{2, 4, 8, 16\}$. In step 4 and step 5 $D(X, C_s)$ was replaced with $(-\log p(X \mid \lambda_s))$ for detailed GMM (DGMM) whereas best matching speaker was selected from the remaining list with $M = \{8, 16, 32\}$ for TIMIT data and, with $M = \{8, 16, 32, 64\}$ for NIST-1999 data, respectively. Detailed description of experimental evaluation of proposed techniques follows in the next chapters.

Chapter 5
Evaluation of Proposed Techniques

5.1 Speech Data Selection for Experiment and Feature Selection

Experiments for this thesis used NIST-1999 speaker recognition evaluation corpus [13] to demonstrate efficiency of techniques proposed for speeding up ASI systems. TIMIT [12] speech data was used to optimize parameters of ASI system. TIMIT speech data consisting of read speech of 630 speakers was down sampled to 8 kHz using anti-aliasing filter. NIST-1999 data is recorded in different sessions. Only data of telephone conversation of 230 male speakers designed for one-speaker detection test was used. The NIST-1999 data was converted from μ-Law companded form to linear PCM format. The selection of speech samples of both TIMIT and NIST-1999 corpora conforms to text independent speaker identification. The sample selection allowed identification test of all registered speakers for both corpora. We used MFCC vectors as feature set for feature level speaker representation like most ASI studies [10].

5.2 Feature Extraction and Speaker Model Generation

Table 5.1 lists details of different parameters used for MFCC feature vector extraction from training and testing speech samples. The speech samples were divided into 30 millisecond long overlapping frames. A ratio of average frame energy was used as threshold to discard silence frames. FFT was computed to get magnitude spectrum of voiced frames after applying Hamming window. Triangular filterbank of selected number of frequencies approximating to Mel-frequency scale was applied to magnitude spectrums using in Equation (5.1). Where f_{Lin} is frequency on linear scale and f_{Mel} is corresponding frequency on Mel scale.

$$f_{Mel} = 2595 \log_{10}(1 + f_{Lin} / 700) \tag{5.1}$$

Table 5.1: Listing of parameters for MFCC feature vector extraction process

Parameter of MFCC vector extraction and speaker models	TCPDE Experiment		HSP Experiment	
Speech corpora	TIMIT	NIST-1999	TIMIT	NIST-1999
Files used for training	'sa','sx'	'A'	'sa','sx'	'A'
Total duration of training sample (S)	22.4	60	22.4	60
Files used for testing	'si'	'B'	'si'	'B'
Total duration of testing sample (S)	8.4	60	8.4	60
Sampling frequency (kHz)	8	8	8	8
Original PCM encoding of speech data	Linear	μ-law	Linear	μ-law
Frame size (millisecond)	30	30	30	30
Frame shift (%)	40	40	66	66
Silence frame energy threshold (%)	15	15	15	15
Training data silence frames (%)	8.91	8.57	9	8
Training data silence frames (%)	5.9	5.5,	6	5.5
Triangular filters (count)	27	31	47	47
First triangular filters ignored (count)	None	3	None	3
Last triangular filters ignored (count)	None	3	None	5
Feature vector size *d* (count)	15	15	17	17

Then DCT of responses of selected triangular filters was taken after compressing them by taking log. Ignoring the first value which only represents bias of input frame to DCT, next *d* values of DCT cepstrum were selected as *d*-dimensional MFCC feature vectors. MFCC vectors were computed once and stored for training speaker models and testing different ASI systems.

For CPDE and TCPDE studies, MFCC vector parameters were optimized, after repeated experimentation, to get 100% accuracy for TIMIT data. For NIST-1999 data, size of triangular filterbank was set 31 after re-experimenting on filter sizes $\{25, 27, 29, 31, 33\}$ for maximum accuracy. The outputs of first and last three filters were ignored while computing MFCC feature vectors of size 15 as shown in Table 5.1. VQ codebooks of sizes $\{2^n \mid 5 \le n \le 10\}$ and $\{2^n \mid 5 \le n \le 11\}$ were trained for CPDE and TCPDE studies using LBG algorithm from TIMIT and NIST-1999 data, respectively. For CSPDE experimentation code vectors were sorted in decreasing order of clusters.

However for HSP studies, first the accuracy of VQ based ASI systems for NIST-1999 data was maximized for MFCC feature vector size $d = 17$ through repeated experimentation for triangular filterbank size range 31 to 51. Outputs of all triangular filters were used for TIMIT data for computing DCT. For NIST-1999 data first three and last five triangular filters were ignored. VQ codebooks of sizes $\{2^n \mid 1 \le n \le 9\}$ and $\{2^n \mid 1 \le n \le 10\}$ were trained through LBG algorithm for TIMIT and NIST-1999 data, respectively. For GMM based HSP studies, VQ codebooks were used as initial guess of mean vectors to EM algorithm to train GMM parameters. GMMs of sizes $\{2^n \mid 1 \le n \le 5\}$ and $\{2^n \mid 1 \le n \le 6\}$ were trained for TIMIT and NIST-1999 data, respectively, to be used as CGMMs and DGMMs.

All algorithms were implemented in Microsoft C# language. The computer programs were run on Windows Vista(TM), installed on HP Compac dx7400 with Intel(R) Core(TM)2

Duo CPU E6550 @2.33 GHz with 2 GB RAM. Speaker identification time of ASI systems was calculated using 'System.DateTime.Now' method of C#.

5.3 Results and Discussions

All close-set ASI systems based on full CCS, PDE, CSPDE, CPDE and TCPDE showed same accuracies, without any degradation, for TIMIT and NIST-1999 speech data as listed in Table 5.2. All registered speakers were tested for accuracy evaluation for both the corpora. There is increase in accuracy for larger codebook size but TIMIT data showed effect of over fitting degradation for codebook size 1024. Hautamaki et al., [61] used GMM-UBM and graph matching techniques to achieve 77% speaker identification accuracy with only 30 registered speakers from NIST-1999 data. They also used 'A' files for training and 'B' files for testing. Our accuracy value of 74.35%, with 230 registered speakers, compares well since accuracy decreases with increase in database size.

Better approximation means earlier hit to closest code vector during CCS which increases the elimination performance of a PDE for speedup. Average behavior of PDE, CSPDE, CPDE and TCPDE in hitting closest code vector versus the scan order of code vectors is depicted in Figure 5.1. Average frequency of hitting the closest code vector each was plotted against the scan sequence order of the code vector being tried in 32 sized codebooks of NIST-1999 data. For 55% of times closest code vector was identified at the very start of CCS with CPDE and TCPDE. For PDE and CSPDE the corresponding frequency is only 4.2% and 3.5% only. The improvement of TCPDE over CPDE is shown in triangular regions in Figure 5.1.

Table 5.2: Accuracy of VQ systems

Codebook Size	Accuracy %	
	TIMIT	NIST-1999
32	87.14	65.65
64	97.30	70.43
128	98.89	71.74
256	99.84	73.91
512	100.00	73.91
1024	99.84	73.48
2048	--	74.35

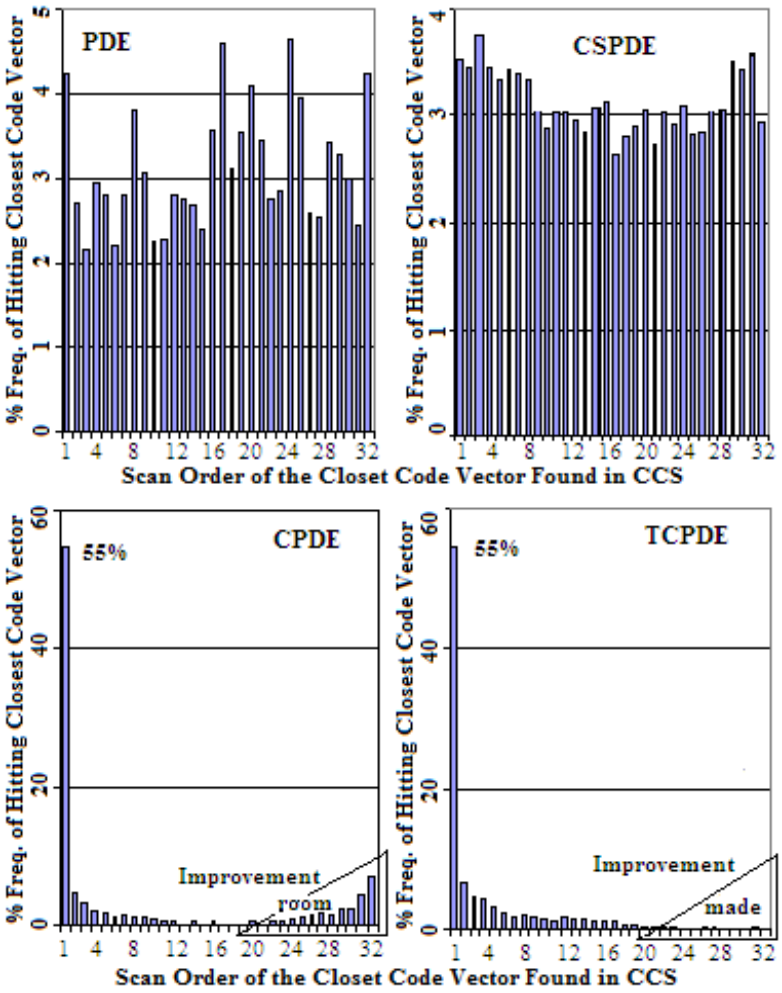

Figure 5.1: Average approximation performances of PDE, CSPDE, CPDE and TCPDE as earlier hitting the closest code vector during CCS in codebooks sized *M*=32 of NIST-1999 data

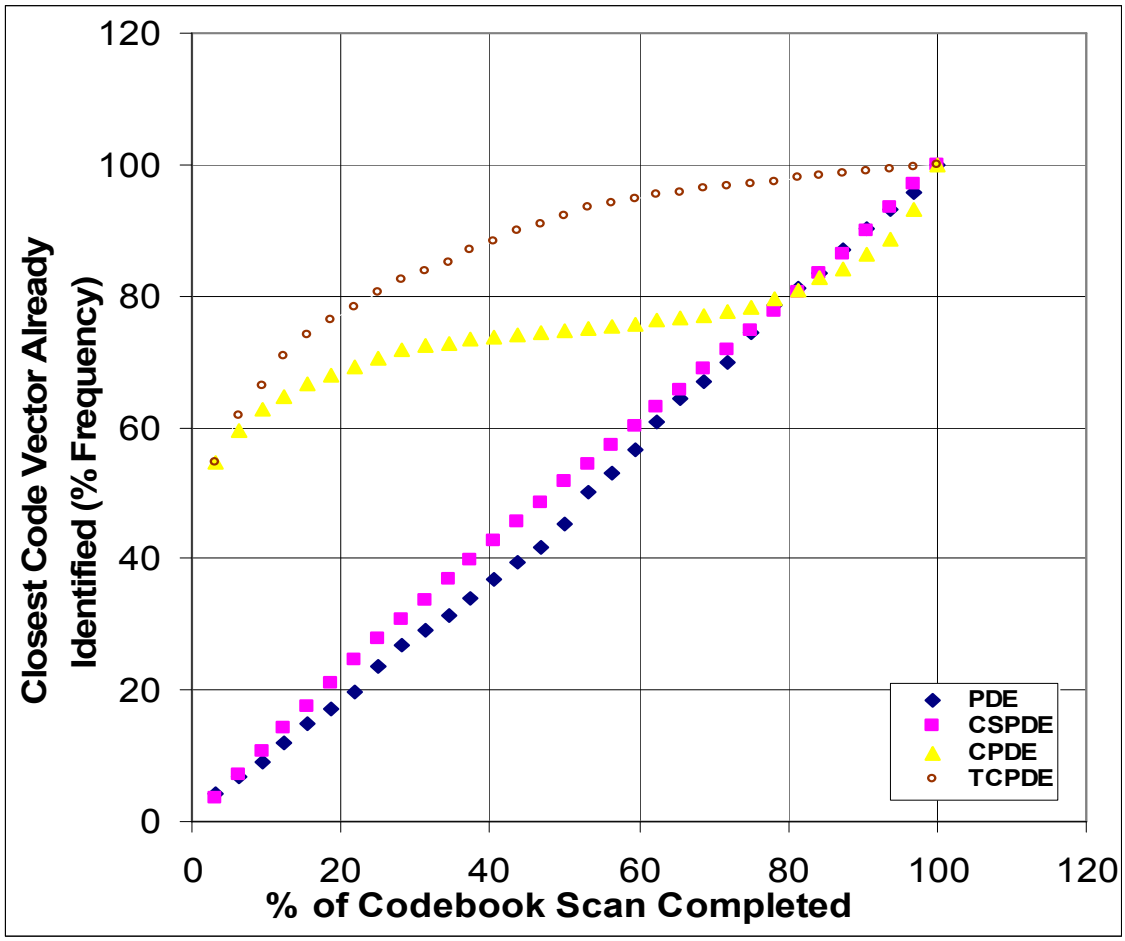

Figure 5.2: Comparison of incremental approximation performance of PDE, CSPDE, CPDE and TCPDE for correct selection of closest code vector along CCS progress through 32 sized codebooks of NIST-1999 data

Figure 5.2 depicts the incremental average performance of TCPDE, CPDE, CSPDE and PDE as CCS progresses through the codebook. The figure shows that TCPDE hits the closest code vector 90% of times before half scan of the codebooks is completed. However, for CPDE, CSPDE and PDE the corresponding values are 74%, 64% and 45%, respectively. The higher frequency values show better 'approximation and elimination' attained by the algorithm that causes greater speedup.

The baseline full search VQ-based ASI systems perform the basic core-steps $dMNT$ times to identify speaker of a test sample. Speedup capability of a PDE variant is proportionate to the reduction in number of executions of the basic core-steps.

Table 5.3 lists the number of executions of the core-steps avoided, on the average, compared to full search CCS for 630 TIMIT and 230 NIST-1999 speakers. The listed values of $(d-d')/d$ were calculated within the program by counting the executions of the basic steps to show approximation and elimination performance of PDE varients. Entropy H and normalized entropy $H/\log_2 M$ of each code vectors are also listed for each codebook size M. Where $H = -\sum_{m=1}^{M} P_m \log_2 P_m$ and, P_m represents the fraction of code vectors in each cluster of the codebook. Consistency in normalized entropies of different sized codebooks of MFCC vectors show a trend contrary to [58]. The ratio of the number of times core-steps are performed by CSPDE, CPDE and TCPDE with respect to PDE decreases for larger codebook size. The CPDE and TCPDE algorithms substantially reduce d' and show greater expected speedup factor as compared to CSPDE. Entropies of code vectors for NIST-1999 data are larger than TIMIT data. But CSPDE results in smaller d' for NIST-1999 than for TIMIT.

Table 5.3: Approximation and elimination performance of PDE variants

Search Algorithm used	TIMIT data			NIST-1999 data		
	$\frac{d-d'}{d}$ %	Expected speedup factor	M / H / $H/\log_2 M$	$\frac{d-d'}{d}$ %	Expected speedup factor	M / H / $H/\log_2 M$
PDE	58.67	1	32	53.91	1	32
CSPDE	69.65	1.36	4.80	70.10	1.54	4.90
CPDE	76.73	1.78		78.08	2.10	
TCPDE	76.97	1.80	0.96	78.30	2.21	0.98
PDE	65.37	1	64	61.00	1	64
CSPDE	75.56	1.42	5.77	75.19	1.57	5.88
CPDE	79.97	1.73		80.83	2.03	
TCPDE	80.28	1.76	0.96	81.08	2.06	0.98
PDE	70.86	1	128	66.95	1	128
CSPDE	79.39	1.41	6.73	79.00	1.57	6.86
CPDE	82.27	1.64		82.74	1.92	
TCPDE	82.61	1.68	0.96	83.02	1.95	0.98
PDE	75.02	1	256	71.69	1	256
CSPDE	82.25	1.41	7.69	81.62	1.54	7.85
CPDE	83.93	1.56		84.21	1.79	
TCPDE	84.28	1.59	0.96	84.42	1.82	0.98
PDE	77.88	1	512	75.43	1	512
CSPDE	84.32	1.41	8.61	83.59	1.50	8.82
CPDE	85.15	1.49		85.27	1.67	
TCPDE	85.49	1.53	0.96	85.58	1.70	0.98
PDE	79.56	1	1024	78.14	1	1024
CSPDE	85.72	1.43	9.46	85.09	1.47	9.77
CPDE	85.97	1.46		86.13	1.58	
TCPDE	86.31	1.49	0.95	86.42	1.61	0.98
PDE	--	--	--	80.00	1	2048
CSPDE				86.24	1.45	10.66
CPDE				86.71	1.51	
TCPDE				87.07	1.55	0.97

The computation cost overhead of conditional branching and managing indices of code vectors was ignored in calculating speedup factors listed in Table 5.3. However, using system clock, actual time of speaker identification and speedup factors with respect to PDE are listed in Table 5.4 to guide re-implementation of proposed techniques in real world ASI systems. To make it independent of feature type, time consumed in feature extraction is not included in speedup computation as shown in Table 5.4.

The speedup factor of CSPDE decreases with codebook size in general. However, the factor is unusually high for codebook size 1024 and low for codebook size 64 for TIMIT and NIST-1999 data, respectively. On the whole speedup factors of CPDE and TCPDE are higher than CSPDE for both corpora. CPDE is up to 37% and 56% faster than PDE for TIMIT and NIST-1999 data respectively.

Figure 5.2 show that CPDE reaches the closest code vector when scanning the later half of the codebook 23% of times. TCPDE reduces this delay by toggling between clockwise and anti-clockwise directions. Thus TCPDE is up to 48% and 74% faster than PDE for TIMIT and NIST-1999 data, respectively. Better performance of CPDE and TCPDE than CSPDE empirically proves the existence of M favorable scan orders that are temporally selectable to benefit from stationarity in speech signal.

Speedup performance shown in Table 5.3 and Table 5.4 did not incorporate vector sequence pruning. In order to compare our framework for ASI speedup with [10], VSP is individually combined with CPDE and TCPDE. Let the two combinations be represented as VSPCPDE and VSPTCPDE, respectively. Table 5.5 lists speedup factors of VSPCPDE and VSPTCPDE with respect to baseline systems that use full search.

Table 5.4: Time based average speedup performance of CSPDE, CPDE and TCPDE compared with PDE

Codebook size	Search algo used	TIMIT data		NIST-1999 data	
		ID Time (S)	Speedup factor	ID Time (S)	Speedup factor
32	PDE	0.60	1	1.48	1
	CSPDE	0.55	1.09	1.24	1.13
	CPDE	0.44	1.37	0.95	1.56
	TCPDE	0.40	1.48	0.85	1.74
64	PDE	1.03	1	2.52	1
	CSPDE	0.96	1.08	2.21	1.05
	CPDE	0.76	1.37	1.65	1.53
	TCPDE	0.71	1.45	1.57	1.61
128	PDE	1.81	1	4.43	1
	CSPDE	1.74	1.04	4.01	1.11
	CPDE	1.35	1.34	2.93	1.51
	TCPDE	1.26	1.43	2.76	1.61
256	PDE	3.24	1	7.81	1
	CSPDE	3.15	1.03	7.39	1.06
	CPDE	2.50	1.29	5.39	1.45
	TCPDE	2.33	1.39	5.06	1.54
512	PDE	6.00	1	14.22	1
	CSPDE	5.80	1.03	13.76	1.03
	CPDE	4.84	1.24	10.37	1.37
	TCPDE	4.48	1.34	9.71	1.46
1024	PDE	11.07	1	25.77	1
	CSPDE	10.48	1.06	25.09	1.03
	CPDE	9.13	1.21	19.83	1.3
	TCPDE	8.45	1.31	18.44	1.4
2048	PDE	--	--	47.19	1
	CSPDE			45.57	1.04
	CPDE			38.24	1.23
	TCPDE			35.23	1.34

Table 5.5: Average Speedup performance of VSPCPDE and VSPTCPDE

Codeook Size	Search algorithm Type	TIMIT data		NIST-1999 data	
		Time (S)	Speedup factor	Time (S)	Speedup factor
32	Baseline	1.40	1	3.19	1
	VSPCPDE	0.40	3.53	0.80	3.97
	VSPTCPDE	0.37	3.77	0.76	4.19
64	Baseline	2.70	1	6.22	1
	VSPCPDE	0.67	4.01	1.38	4.52
	VSPTCPDE	0.62	4.37	1.3	4.78
128	Baseline	5.25	1	12.18	1
	VSPCPDE	1.19	4.41	2.44	4.98
	VSPTCPDE	1.10	4.77	2.29	5.32
256	Baseline	10.52	1	24.16	1
	VSPCPDE	2.18	4.83	4.47	5.41
	VSPTCPDE	2.01	5.23	3.9	6.20
512	Baseline	21.08	1	48.37	1
	VSPCPDE	4.17	5.05	8.58	5.64
	VSPTCPDE	3.83	5.50	8.01	6.04
1024	Baseline	42.19	1	95.93	1
	VSPCPDE	7.87	5.36	16.41	5.85
	VSPTCPDE	7.28	5.79	15.3	6.27
2048	Baseline			191.18	1
	VSPCPDE	--	--	31.50	6.07
	VSPTCPDE			29.11	6.57

The speedup factors listed in Table 5.5 for NIST-1999 data are greater than those for TIMIT data for corresponding codebook size. VSPCPDE and VSPTCPDE did not degrade and had same accuracies as given in Table 5.2 for corresponding corpora and the codebook size. The speedup factors of VSPCPDE and VSPTCPDE in Table 5.5 are double or better than those for VPT combined with speaker pruning as reported [10].

Kinnunen et al., [10] did not get significant results from their VQ-based HSP experimentation and consequently ignored HSP for GMM based experiments altogether. Factors other than the sizes of coarse and detailed speaker models need to be discovered to highlight the significance of HSP. To get greater speedup from HSP, core objective is to prune out larger number of unlikely speaker through ranking with as smaller sized coarse model as possible. Choice of CBD as distance measure was tried and highly significant speaker ranking results were obtained for coarser codebooks.

Table 5.6 summarizes performance indicators namely identification error, ranking time and standard deviation of ranking (SDR) for coarse codebooks of sizes 2, 4, 8, 16, and 32. Figure 4.2 depicts detailed comparison between ranking order of EUD and CBD based searches of true speaker using PDE for 630 TIMIT speakers.

Identification error of CBD based search is less than EUD based search for each coarse codebook by 2 to 6 times. Standard deviation of ranking order of CBD based search is 2 to 16 times smaller than that of EUD based search. Smaller value of SDR indicates possibility of pruning larger number of speakers before searching through detailed codebooks.

Best testing results of baseline full search, PDE search and PDE+HSP based search using coarse codebook of size 4 are summarized in Table 5.7 for TIMIT and NIST-1999 data. No over fitting effect is observed in VQ-based systems for both the corpora.

Table 5.6: Ranking order evaluation indicators for EUD-PDE and CBD-PDE based VQ systems on TIMIT data

Model size	EUD Based			CBD Based		
	Error %	Time (S)	SDR	Error %	Time (S)	SDR
2	97.30	0.03	128.59	34.13	0.28	11.57
4	66.19	0.08	23.44	29.05	0.18	9.40
8	76.83	0.06	69.60	34.13	0.50	5.75
16	41.11	0.16	6.58	19.21	0.31	5.75
32	45.24	0.11	24.38	7.62	0.89	1.50

Table 5.7: Best speedup results of VQ (PDE+HSP) based systems on TIMIT and NIST-1999 data

VQ system		Best TIMIT results			Best NIST-1999 results		
Model size	Search type	Error %	Time (S)	Speedup factor	Error %	Time (S)	Speedup factor
64	Baseline	2.70	1.79	1:1	29.57	4.18	1:1
	PDE	2.70	0.57	3.16:1	29.57	1.58	2.65:1
	PDE+HSP	2.70	0.20	9.01:1	29.57	0.72	5.77:1
128	Baseline	1.27	3.55	1:1	25.22	8.20	1:1
	PDE	1.27	1.03	3.45:1	25.22	2.76	2.98:1
	PDE+HSP	1.27	0.22	16.06:1	25.22	0.82	10.00:1
256	Baseline	0.63	7.21	1:1	23.48	16.47	1:1
	PDE	0.63	1.90	3.80:1	23.48	4.91	3.35:1
	PDE+HSP	0.63	0.32	22.46:1	23.48	0.99	16.63:1
512	Baseline	0.63	14.32	1:1	24.35	32.64	1:1
	PDE	0.63	3.59	3.99:1	24.35	9.09	3.59:1
	PDE+HSP	0.63	0.73	19.75:1	24.35	1.53	21.40:1
1024	Baseline				24.35	65.77	1:1
	PDE	--	--	--	24.35	16.76	3.92:1
	PDE+HSP				24.35	1.89	34.78:1

Table 5.8: Best speedup results of GMM HSP based systems on TIMIT and NIST-1999 data

GMM system		Best TIMIT results			Best NIST-1999 results		
Model size	Search type	Error %	Time (S)	Speedup factor	Error %	Time (S)	Speedup factor
8	Baseline	2.22	0.84	1:1	30.87	1.90	1:1
	HSP	2.22	0.27	3.16:1	30.87	1.69	1.12:1
16	Baseline	1.27	1.69	1:1	26.52	3.76	1:1
	HSP	1.27	0.27	6.10:1	26.52	2.03	1.85:1
32	Baseline	1.11	3.33	1:1	25.22	7.49	1:1
	HSP	1.11	0.32	10.40:1	25.22	2.13	3.52:1
64	Baseline	--	--	--	26.52	15.01	1:1
	HSP				26.52	2.27	6.61:1

Table 5.9: Best speedup factor of VQ (PDE+HSP) based systems compared with full search for different coarse codebook sizes on TIMIT and NIST-1999 data

DCB size	Best TIMIT speedup factor of HSP compared with full search (best *W* value) for CCB sizes					Best NIST-1999 speedup factor of HSP compared with full search (best *W* value) for CCB sizes				
	32	16	8	4	2	32	16	8	4	2
64	1.92 (616)	3.24 (595)	4.11 (479)	9.01 (590)	6.32 (433)	2.10 (228)	3.72 (225)	5.77 (212)	4.98 (160)	4.56 (129)
128	3.77 (433)	5.49 (616)	5.80 (557)	14.97 (585)	16.06 (557)	4.05 (226)	6.93 (222)	10.00 (212)	6.73 (160)	5.58 (129)
256	7.47 (616)	9.58 (557)	8.27 (450)	22.46 (581)	10.07 (433)	7.90 (225)	12.38 (219)	16.63 (212)	8.69 (160)	6.71 (129)
512	13.23 (603)	7.53 (557)	11.74 (478)	19.75 (538)	11.26 (433)	13.74 (220)	21.40 (219)	11.80 (177)	4.06 (40)	5.70 (92)
1024	--	--	--	--	--	28.09 (225)	34.78 (219)	34.04 (212)	11.88 (160)	8.51 (129)

Table 5.10: Best speedup factor of GMM HSP based systems compared with full search for different coarse GMM sizes for TIMIT and NIST-1999 data

DGMM size	Best TIMIT speedup factor of HSP compared with full search (best *W* value) for CGMM sizes				Best NIST-1999 speedup factor of HSP compared with full search (best *W* value) for CGMM sizes			
	16	8	4	2	16	8	4	2
8	--	--	1.73 (608)	3.16 (610)	--	--	1.12 (150)	*2.61*
16	--	1.94 (626)	3.24 (608)	6.10 (610)	--	1.85 (225)	1.61 (150)	*4.01*
32	1.94 (628)	3.67 (628)	5.77 (608)	10.40 (610)	1.80 (226)	3.52 (225)	2.04 (150)	*3.49*
64	--	--	--	--	3.51 (226)	6.61 (225)	2.37 (150)	*6.85*

Best testing results of baseline full search and HSP for different sizes of detailed GMM and some sizes of coarse GMM are summarized in Table 5.8 for TIMIT and NIST-1999 data. The results in Table 5.8 show that identification error decreases with increase in GMM size. However, slight over fitting effect is observed for NIST-1999 data but not for TIMIT data. For NIST-1999 error rate of 23.48% and 25.22% for 230 registered speakers for our VQ and GMM based systems, respectively, are better than error reported for 30 registered speakers in [61].

Listings of best speedup results for with PDE+HSP using coarse codebooks sized 4, 8, 16 and 32 are given in Table 5.9 for TIMIT and NIST-1999 data, at maximum value of W that did not degrade the accuracy. All HSP speedup results shown in Table 5.9 are better than corresponding PDE search results which shows added advantage of HSP, except for coarse codebooks sized 32 for detailed codebook sized 64.

Listing of best speedup results with HSP using coarse GMM models sized 2, 4, 8 and 16 is given in Table 5.10 for TIMIT and NIST-1999 data, at maximum value of W . The accuracy did not degrade except for NIST-1999 data with coarse GMM size 2. That is why the corresponding column in Table 5.10 is listed without any W value in italic. Somehow, speaker number 179 that is correctly identified by detailed GMMs is ranked at 189 by coarse GMM sized 2. This situation limited the pruning factor W to be less than 41 and resulted in slow speed of HSP. However, the speedup factors of CGMM sized 2 shown in italics in Table 5.10 are with accuracies compromised by 0.43%.

Figure 5.3 and Figure 5.4 depict HSP performance details on TIMIT data for different coarse and detailed models at different values of W of VQ and GMM based ASI systems studied, while Figure 5.5 and Figure 5.6 show detailed HSP performance for NIST-1999 data.

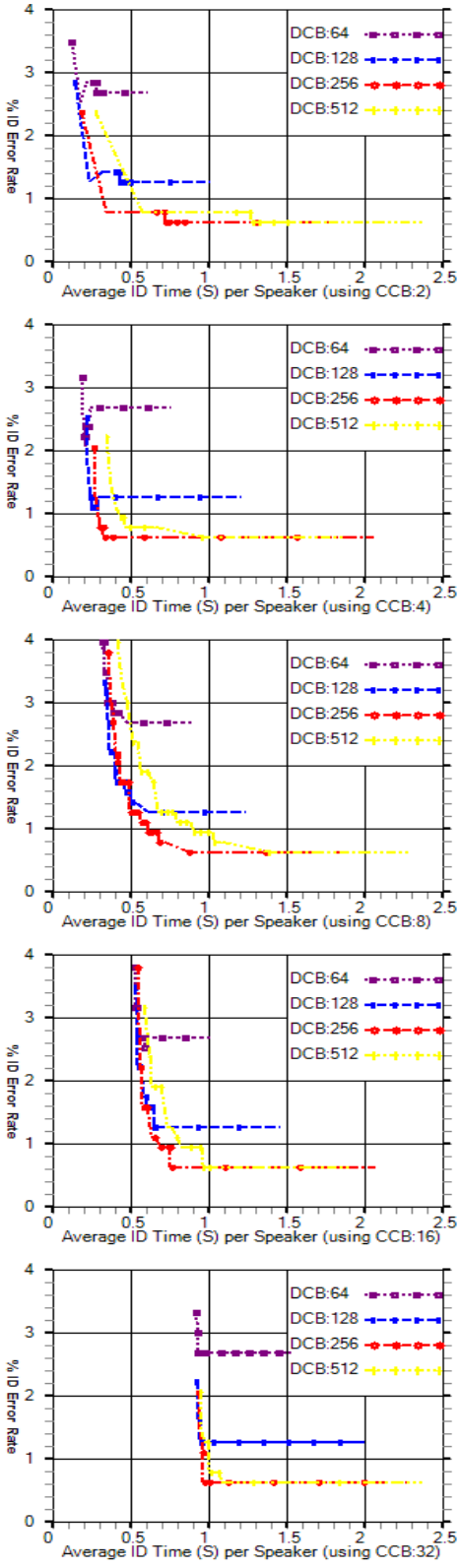

Figure 5.3: HSP performance parameters for VQ for TIMIT data

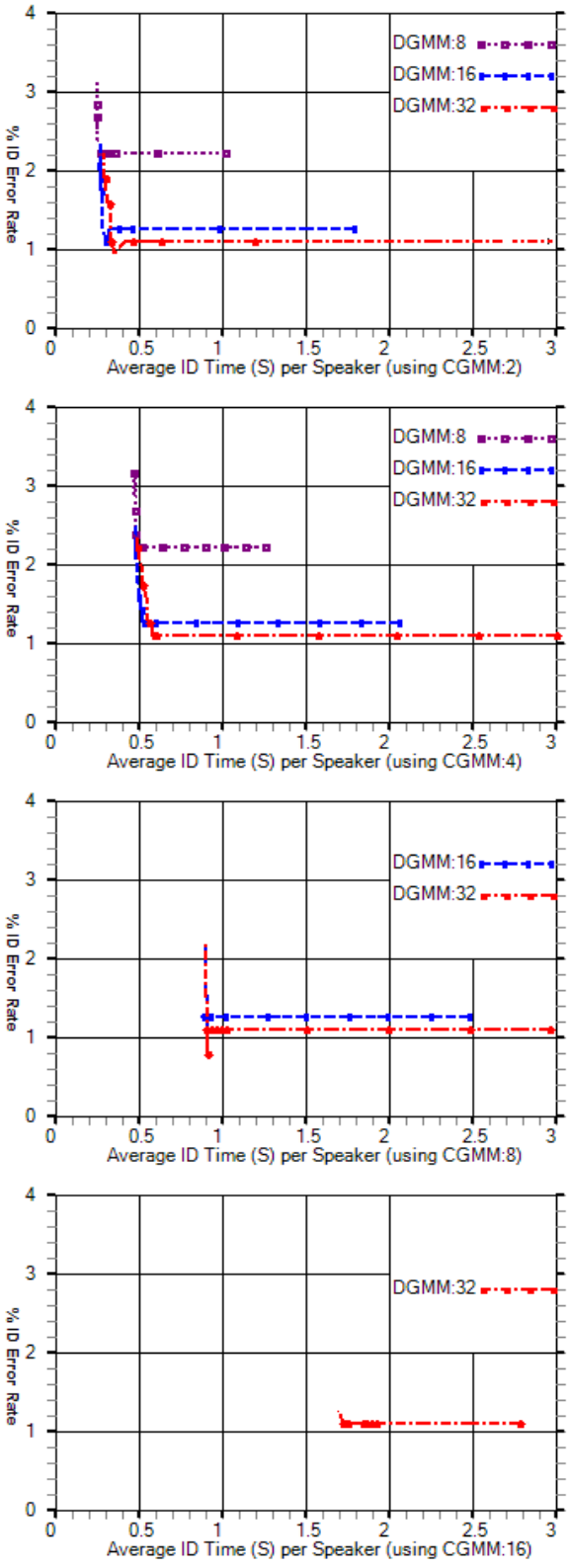

Figure 5.4: HSP performance Parameters for GMM for TIMIT data

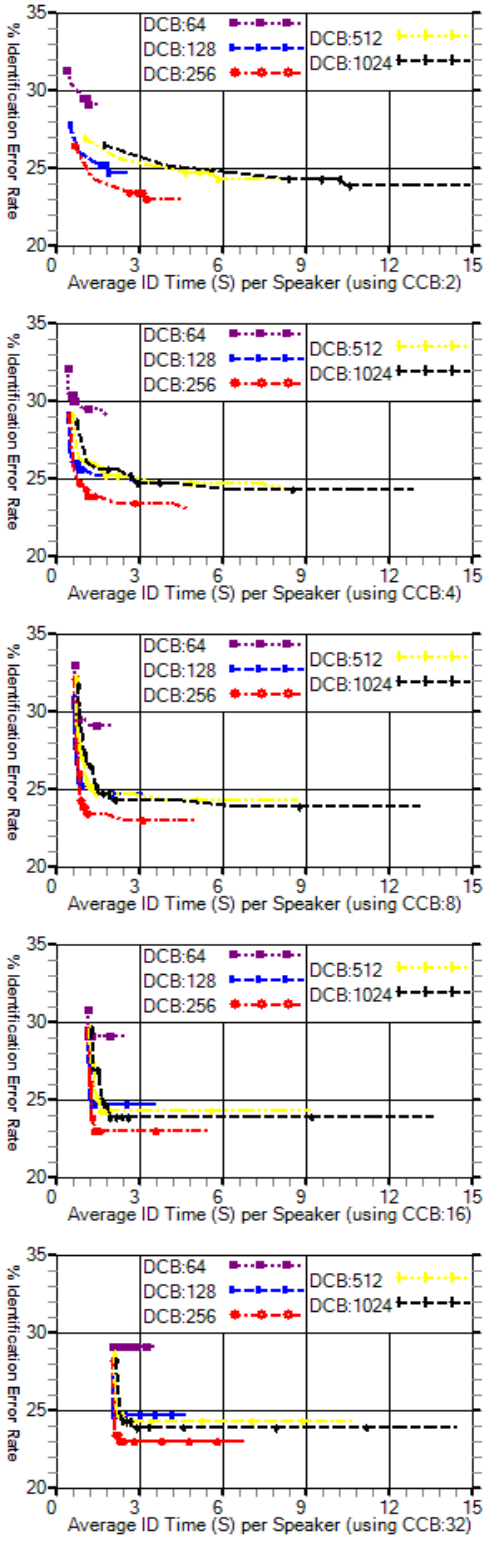

Figure 5.5: HSP performance parameters for VQ for NIST-1999 data

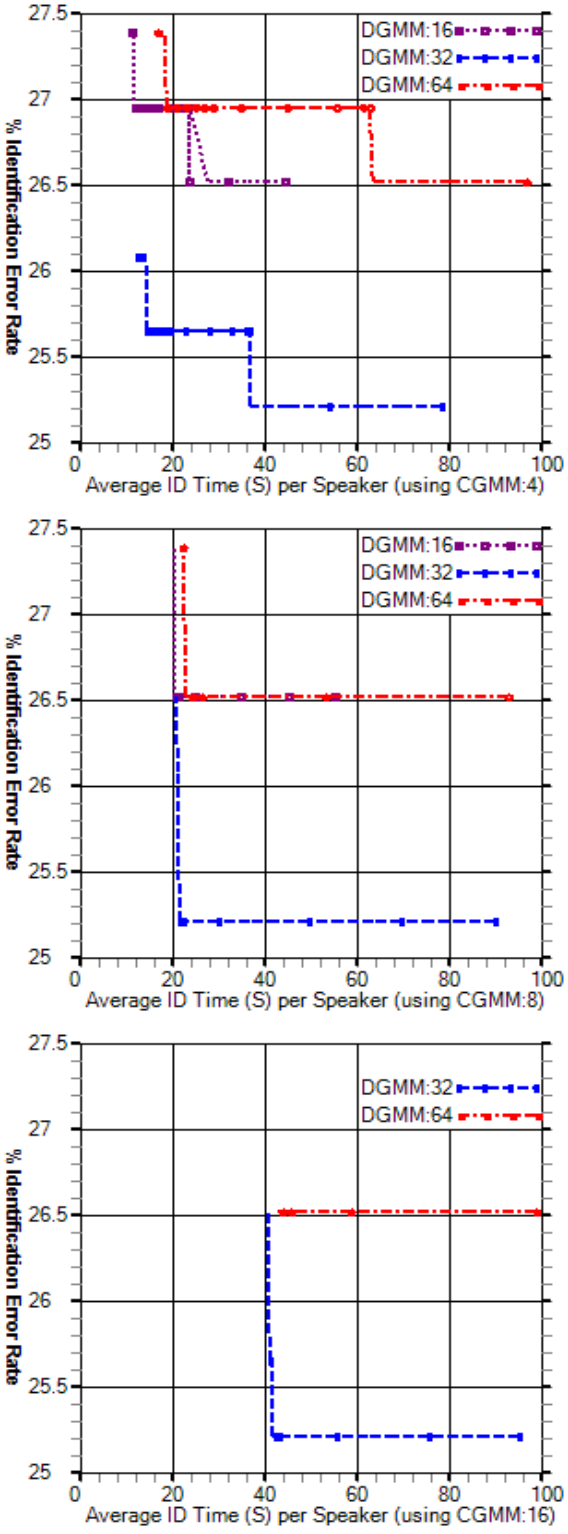

Figure 5.6: HSP performance parameters for GMM for NIST-1999 data

The horizontal portion from right to left of each graph for a particular pair of CCB and DCB in Figures 5.3 and 5.4 and a particular pair of CGMM and DGMM in Figures 5.5 and 5.6 shows room for speedup through HSP with larger W value to prune out more unlikely speakers without any accuracy loss. The graphs which suddenly turn upwards indicate sharp loss in accuracy with undue larger value of W . However, the graphs that smoothly increase curve upwards indicate gradual loss in accuracy for further increase in W value.

In the end we comment on differences between TCPDE and HSP studies. The CPDE and TCPDE experimentation was conducted with high time resolution by having smaller shift in frames which increased the number of training and testing MFCC vectors of size $d = 15$. Other MFCC feature extraction parameters were then optimized to achieve 100% accuracy for TIMIT data. HSP experimentation was conducted at lower time resolution just to decrease the size of X in order to reduce speaker identification time. However, feature vector size was increased to $d = 17$ in order to increase accuracy for NIST-1999 data. The rest of the feature extraction parameters were optimized to get accuracy of VQ-based ASI systems comparably better than [61] for NIST-1999 data. However, this change resulted in decreased speaker identification accuracy for TIMIT data which consist of smaller speech samples than NIST-1999.

Table 5.11: Summary of improvements in ASI speedup compared with existing work

ASI Aspect Compared	Speedup Technique	Proposed	*Max Speedup Factor	Number of Registered Speakers (N)	Basis of Speedup Factor	Accuracy Degrade
CCS Speedup	$CSPDE	By Paliwal and Ramasubramanian [58]	1.13	230	Plain PDE	No
	CPDE	In this thesis	1.56	230	Plain PDE	No
	TCPDE	In this thesis	1.74	230	Plain PDE	No
VQ-based ASI Speedup	VSPCPDE	In this thesis	6.07	230	Baseline	No
	VSPTCPDE	In this thesis	6.57	230	Baseline	No
	(Testing time) HSP	By Kinnunen et al., [10]	1.51	230	Baseline	~Depends on K and W
	(Testing time) HSP	In this thesis	8.87	230	Baseline	~Depends on K and W
	HSP+PDE	In this thesis	34.78	230	Baseline	~Depends on K and W
	HSP+PDE	In this thesis	22.46	630	Baseline	~Depends on K and W
	#PQP	By Kinnunen et al., [10]	9	230	Baseline	Depends on W and Quantization Level
GMM-based ASI Speedup	(Testing time) HSP	In this thesis	10.40	630	Baseline	~Depends on K and W
	(Testing time) HSP	In this thesis	6.61	230	Baseline	~Depends on K and W
	UBM/GMM (Training time) HSP	By McLaughlin et al., [9]	2	20	Baseline	Depends on Similarity of GMM states
	ISODATA Algo (Training time) HSP	By Sun et al., [8]	6	160	Baseline	Depends on Hierarchical Structure

$ Propose in [58] for speech compression but tested first time for ASI in this thesis.
* Without accuracy loss compared with corresponding baseline.
~ Depends on K and W for a particular M of DCB or DGMM.
# Pre-quantization and static speaker pruning [10].

Chapter 6

Conclusions

Techniques for speeding up VQ-based and GMM-based real-time speaker identification without loss of accuracy have been presented. Departing from Voronoi view, refer for example [60], of VQ codebook, proximity trend ingrained into code vector indexes generated by LBG has been utilized to propose CPDE that makes a circular scan through the codebook. In this circular scan, selection of code vector that proved closest in the previous CCS is proposed to be the first candidate for subsequent CCS. For VQ-based systems, stationarity in speech signal has been capitalized to substantially enhance partial elimination as compared to native PDE and CSPDE. It has been observed that CPDE is faster than native PDE by a margin of up to 56% for NIST-1999 data and up to 37% for TIMIT data.

To further improve approximation, the delay occurring in the approach to the closest code vectors existing on the opposite direction of CPDE scan is reduced by proposed TCPDE that toggles between anti-clockwise and clockwise directions. TCPDE is faster than native PDE by a margin of up to 74% for NIST-1999 data and up to 48% for TIMIT data.

After restating CSPDE of [58], Ramasubramanian and Paliwal state their observation about LBG in [57] as, "*The codebooks obtained at the end of the training process using the clustering algorithms such as the LBG algorithm have arbitrary orderings and are not guaranteed to be arranged in the favorable order.*" On the contrary, faster speeds of TCPDE and CPDE over CSPDE empirically show the existence of M favorable scan sequences that are temporally selectable to speedup CCS through utilization of stationarity in speech signal.

VSPTCPDE and VSPCPDE have been proposed by combining VSP with TCPDE and CPDE, respectively. The proposed combinations prune test feature vector sequence for unlikely

codebooks. For NIST-1999 and TIMIT data speedup factors achieved by VSPCPDE on the average are up to 6.07 and 5.36 as compared to baseline full search, respectively. The speedup factors of VSPTCPDE for TIMIT and NIST-1999 data are up to 5.8 and 6.6 as compared to baseline full search, respectively.

Improvement in HSP algorithm to speedup speaker identification is also explored in this thesis. Kinnunen et al., [10] observed that HSP increased speed of VQ-based ASI less than adaptive speaker pruning and hence they completely neglected HSP for GMM based ASI studies. Detailed study of HSP for GMM based ASI systems has been presented in this thesis. Experimental results presented herein show that GMM based HSP prunes a large number of speakers from the candidate list to achieve speedup factors of up to 6.61 and 10.40 for 230 NIST-1999 speakers and 630 TIMIT speakers, respectively, with no accuracy loss.

For VQ-based ASI systems, CBD based PDE has been observed to give higher accuracy and hence better ranking than EUD based PDE for codebooks sized (<64). Therefore, CBD allows use of smaller coarse codebooks. Thus CBD based PDE ranking of registered speakers with coarse codebooks is capitalized to prune a large number of speakers. Combination of HSP and PDE has achieved speedup factors up to 22.46 and 34.78 for 630 TIMIT speakers and 230 NIST-1999 speakers, respectively, with no accuracy loss.

Chapter 7

Future Directions

Most of the ASI systems use MFCC or LPCC feature vectors for speaker modeling. Other features set must also be investigated that increase accuracy with smaller speaker models. FFT was emphasized when digital computers were slow. DFT obviates $2^n \,|\, n \in \mathbb{N}$ a frame size constraint imposed by FFT. Therefore, DFT that can avoid zero padding at speech frames needs be re-evaluated on digital systems to increase accuracy as it DFT avoids addition of spurious frequencies that are included in the spectrum by FFT.

Identification accuracy of an ASI system increases with the parameters d and M to be defined at design time of speaker model. The number of voice feature selected is defined by d while the variation in the features is represented through M values of features and other similarity representing parameters. The accuracy decreases with increase in the number of registered speakers N. The length of X represented by T improves thoroughness in characterizing the speaker and, increases the quality of matching and the accuracy of an ASI system. However, increase in any of the four parameters increases the identification time. The length of $\tilde{X}$ represented by $\tilde{T}$ allows better representation of variations in the speaker's features to be modeled and improves accuracy which can indirectly serve to reduce d or M and hence increase speed of ASI system at comparable accuracy.

Over fitting effect is generally seen with increase in speaker model size M which is doubled to get the next detailed model through LBG or other clustering algorithm. There is also need to search for methods that allow increasing model size for more details without the

constraint of $M = 2^n \,|\, n \in \mathbb{N}$. Over lifting this constraint would allow optimum selection of M with optimum accuracy and speed.

Importance of reducing computations for increased efficiency can not be over emphasized. So far machine listeners for ASI have been modeled based on human speech production and human speech perception systems. Firstly there is need to decouple ASI from speech processing methods which include processing to serve information tapping of intermediate data for human perception. Success in eliminating such processing steps from speaker recognition can be later also utilized to speedup speech processing.

References

[1] Campbell, J. (1997). Speaker Recognition: A Tutorial: Proc., of IEEE, 85(9), 1437-1462

[2] Quatieri, F.T., (2002). Discrete-time Speech Signal Processing Principles and Practice: Pearson Education.

[3] Markel J., Oshika B., and Gray, J., (1977). Long-term features averaging for speaker recognition: IEEE Trans. Acoustics, Speech, and Signal Processing, 25(4), 330-337.

[4] Prabhakar, S., Pankanti, S., (2003). Biometric Recognition: Security and Privacy Concerns, IEEE Security and Privacy Magazine, (1), 33-42

[5] Glaeser A., and Bimbot F., (1998). Steps Towards the Integration of Speaker Recognition in Real-world Telecom Applications: In Proc. Int. Conference on Spoken Laguage Processing, (ICSLP) Sydney, NSW, Australia.

[6] Jia W., and Chan W. Y., (2000). An Experimental Assessment of personal Speech Coding: Speech Communication. 30 (1) 1-8.

[7] Reynolds D., Quatieri F. T., and Dunn R., (2000). Speaker Verification Using Adapted Gaussian Mixture Models: Digital Signal Processing, 10(1), pp-19-41.

[8] Sun B., Liu W., and Zhong Q., (2003). Hierarchical Speaker Identification Using Speaker Clustering. In Proc. Int. Conference on Natural Language Processing and Knowledge Engineering. 299–304.

[9] McLaughlin, J., Reynolds, D., and Gleason, T., (1999). A Study of Computation Speed-ups of the GMM-UBM Speaker Recognition System: In Proc. 6th European Conference on Speech Communication and Technology (Eurospeech), 1215–1218.

[10] Kinnunen T., Karpov E., and Franti P., (2006). Real-Time Speaker Identification and Verification: IEEE Transactions on Audio, Speech and Language Processing, 14(1), 277-288.

[11] Cole R., Noel M., Noel V., (1998). The CSLU Speaker Recognition Corpus: Proc 5th Int. Conference on Spoken Language Processing (ICSLP), Sydney, Australia, pp.3167-3170.

[12] Garofolo J., Lamel L., Fisher W., Fiscus J., Pallett D., Dahlgren N. and Zue V., (1993). TIMIT Acoustic-Phonetic Continuous Speech Corpus.

[13] Martin A., and Przybocki M., (2000). The NIST 1999 Speaker Recognition Evaluation- An Overview: Digital Signal Process. (10) 1-18.

[14] Alexandre A., Botti F., Dessimoz D., and Drygajlo A., (2004). The Effect of Mismatched Recording Conditions on Human and Automatic Speaker Recognition in Forensic Applications: Forensic Science International 146S, 95-99,

[15] Gonzalez-Rodriguez J., Garcia-Gomar M., Ramos-Castro, D., and Ortega-Garcia, J., (2003). Robust Likelihood Ratio Estimation in Bayesian Forensic Speaker Recognition: In Proc. 8th European Conference on Speech Communication and Technology, 693–696.

[16] Pfister B., and Beutler R., (2003). Estimating the Weight of Evidence in Forensic Speaker Verification: In Proc. 8th European Conference on Speech Communication and Technology, 701–704.

[17] Niemi-Laitinen T., Saastamoinen J., Kinnunen T., and Franti P., (2005). Applying MFCC-based Automatic Speaker Recognition to GSM and Forensic Data. In Proc. 2nd Baltic Conference on Human Language Technologies., 317–322.

[18] Thiruvaran, T., Ambikairajah, E., and Epps, J. (2008) FM Features for Automatic Forensic Speaker Recognition. In Proc. Interspeech., 1497–1500.

[19] Ben-Zeghiba, M., and Bourland, H., (2003). On the Combination of Speech and Speaker Recognition. In Proc. 8th European Conference on Speech Communication and Technology., 1361– 1364.

[20] Schmidt-Nielsen, A. and Crystal, T., (1998). Speaker Verification by Human Listeners: Experiments Comparing Human and Machine Performance using the NIST-1998 Speaker Evaluation Data. Digital Signal Processing, (10) 249-266.

[21] Kinnunen, T., and Li. H., (2010). An Overview of Text-Independent Speaker Recognition: from Features to Supervectors: Speech Communication, Elsevier, 52, (1) 12-40.

[22] Soong F.K., Rosenberg A.E., Juang B.H., and Rabiner L.R. A., (1987). Vector Quantization Approach to Speaker Recognition: AT & T Technical Journal, 66, 14-26.

[23] Reynolds D., and Rose R., (1995). Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models: IEEE Trans. on Speech and Audio Processing (3).

[24] Wolf J., (1972). Efficient Acoustic Parameters For Speaker Recognition: Journal of the Acoustic Society of America, 51(6), (Part 2), 2044-2056.

[25] Rose P., (2002). Forensic Speaker Identification: Taylor & Francis, London.

[26] Gopalan K., Mahil S., (1991). Speaker Identification via Singular Value Decomposition of Speech Parameters: Midwest Symposium on Circuits and Systems, IEEE; Vol. 5, pp.725-728

[27] Säonmez M., Shriberg E., Heck L., and Weintraub M., (1998). Modeling Dynamic Prosodic Variation For Speaker Verification: In Proc. Int. Conf. on Spoken Language Processing (ICSLP), 0920.

[28] Plumpe M., Quatieri F. T., and Reynolds D., (1999). Modeling of the Glottal Flow Derivative Waveform with Application to Speaker Identification: IEEE Trans on Speech and Audio Processing 7(5), 569-586.

[29] Tomi K., Ville H., and Franti P., (2003). On the Fusion of Dissimilarity-Based Classifiers for Speaker Identification: In Proc. EUROSPEECH–GENEVA.

[30] Liu L., He J., and Palm G., (1996). Signal modeling for speaker Identification: In Proc. IEEE Int. Conf., on Acoustics, Speech, and Signal Processing, New York USA. (2) 665-668.

[31] Bogdan S., and Gavat I., (2000). Speaker Identification using Discriminative Feature Selection- A Growing Neural Gas Approach: NEUREL, Yugoslavia.

[32] Atal B., (1972). Automatic Speaker Recognition based on Pitch Contours: Journal of the Acoustic Society of America, 52(6) 1687-1697.

[33] Ashour G., and Gath I., (1999). Characterization of Speech during Imitation: Int. Proc 6th European Conf. on Speech Communication and Technology, Budapest, Hungary. 1187-1190.

[34] Campbell, W., Sturim, D., and Reynolds, D., (2006) Support Vector Machines using GMM Supervectors for Speaker Verification: IEEE Signal Processing Letters. 13(5) 308–311.

[35] Jain A., and Zongker D., (1997). Feature Selection: Evaluation, Application, and Small Sample Performance: IEEE Trans on Pattern Analysis and Machine Intelligence (19)153-158.

[36] Sambur M., (1975). Selection of Acoustic Features for Speaker Identification: IEEE Trans Acoustics, Speech, and Signal Processing. 23(2), 176-182.

[37] Cheung R., and Eisenstein B., (1978). Feature Selection via Dynamic Programming for Text-Independent Speaker Identification: IEEE Trans on Acoustics, Speech and Signal Processing (26), 397-403.

[38] Charlet D., and Jouvet, D., (1997). Optimizing Feature Set for Speaker Verification: Pattern Recognition Letters (18), 873-879.

[39] Duda R., Hart P., and Storks D., (2000). Pattern Classification: Wiley Interscience, New York, (2nd ed).

[40] Fukunaga K., (1990). Introduction to Statistical Pattern Recognition: Academic Press, London, (2nd ed).

[41] Fan N., Rosca J., (2003). Enhanced VQ-Based Algorithms for Speech Independent Speaker Identification: In Proc Audio and Video Based Biometric Authentication (AVBPA), Guildford, UK, 470-477.

[42] Desch D., and King R., (1997). Speaker Models Designed from Complete Data Sets: A New Approach to Text-independent Speaker Verification: In Proc. 5th European Conf. on Speech Communication and Technology, Rhodes, Greece, 2323-2326.

[43] Linde Y., Buzo A., and Gray R.M., (1980). An Algorithm for Vector Quantizer Design: IEEE Transactions on Communications, 28(1), 84-95.

[44] Sakoe H., and Chiba S., (1978). Dynamic Programming Algorithm Optimization for Spoken Word Recognition: IEEE Transactions on Acoustics, Speech, and Signal Processing (ASSP), 1(26), 43-49.

[45] He J., Liu L., and Palm G., (1999). A Discriminative Training Algorithm for VQ-Based Speaker Identification: IEEE Trans. on Speech and Audio Processing, 7(3), 353-356.

[46] Higgins A., Bahler L., and Porter J., (1993). Voice Identification using Nearest-Neighbor Distance Measure: In Proc. Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), USA, 375-378.

[47] Reynolds D., and Rose R., (1995). Robust Text-Independent Speaker Identification using Gaussian Mixture Speaker Models: IEEE Trans. Speech Audio Processing, 1(3), 72–83.

[48] Bishop C. (2006). Pattern Recognition and Machine Learning. Springer Science+Business Media, LLC, New York.

[49] Bimbot F., Magrin-Chagnolleau I., and Mathan L., (1995). Second-order Statistical Measures for Text-independent Speaker Identification: Speech Communication, 1-2 (17), 177-192.

[50] Younes B., (1992). Text Independent Talker Identification System Combining Connectionist and Conventional Models: In Proc. Neural Networks for Signal Processing; IEEE-SP Workshop, pp.131-138

[51] Clarkson T., Christodoulou C., Guan Y., Gorse D., Romano A., and Taylor J., (2001). Speaker Identification for Security Systems using Reinforcement –Trained pRAM Neural Networks: IEEE Transaction on Systems, Man and Cybernetics, 1 (31) 65-76.

[52] Vincet W., Campbell W., (2000). Support Vector Machines for Speaker Verification and Identification: In Proc IEEE Signal Processing Society Workshop, 2, 755-784.

[53] Campbell W., Assaleh K., and Broun C., (2002). Speaker Recognition with Polynomial Classifiers: IEEE Trans. on Speech and Audio Processing, 4(10), 205-212.

[54] Afzal M., and Haq S., (2010). Accelerating Vector Quantization Based Speaker Identification, Journal of American Science. 6 (11), 1046-1050

[55] Cheng D., Gersho A., Ramamurthi B., and Shoham Y., (1984). Fast Search Algorithms for Vector Quantization and Pattern Matching: In Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1, pp.9.11.1-9.11.4

[56] Bei H., and Gray R., (1985). An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization: IEEE Transactions on Communication. 33(10) 1132-1133

[57] Ramasubramanian V., and Paliwal K., (2000). Fast Nearest-Neighbor Search Algorithms Based on Approximation-Elimination Search: Pattern Recognition. 33(9) 1497-1510

[58] Paliwal K., and Ramasubramanian V., (1989). Effect of Ordering the Codebook on the Efficiency of the Partial Distance Search Algorithm for Vector Quantization: IEEE Transactions on Communications. 37(5) 538-540

[59] Beigi H., Maes S., Chaudhari U., and Sorensen J., (1999). A Hierarchical Approach to Large-Scale Speaker Recognition: In Proc. Sixth European Conference on Speech Communication and Technology. 2203–2206.

[60] Salomon D., (2007). Data compression: the complete reference, Volume 10, 4th Ed, Springer.

[61] Hautamäki V., Kinnunen T., and Fränti P., (2008). Text-Independent Speaker Recognition Using Graph Matching, Pattern Recognition Letters. 29(9) 1427-1432