PARALLEL IMPLEMENTATION OF A VQ-BASED TEXT-INDEPENDENT SPEAKER IDENTIFICATION

by

Ruhsar Soğancı

B.S., in Computer Engineering, Marmara University, 2001

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2004

ACKNOWLEDGEMENTS

I would like to express my thanks to my supervisors, Professor Fikret Gürgen and Assoc. Prof. Haluk Topcuoglu. Professor Fikret Gürgen directed me with his excellent knowledge on speech based systems, and Assoc.Prof. Haluk Topcuoğlu helped me a lot regarding parallel programming issues. I was very fortunate to have their guidance while preparing my thesis.

Also, I would like to give thanks to ASMA lab members: Assoc. Prof. Can Özturan, A.Burak Gürdağ and A.Haydar Özer for their self-sacrificing efforts to manage the ASMA cluster in Boğaziçi University. I'm thankful to Berna Karakullukçu, who provided me with a serial speaker verification system implemented by using Matlab. Additionally, I whould like to thank my friends Onur Uztüre, Bahar Karaoğlu and Nihal Yıldırım for their supports, Laurie Chandler for editing this text. Also I want to thank Özgür Zan for his encouragement.

This thesis is dedicated to my best friend Tansel Haliç, who provided me with excellent Linux support, and makes me smile whenever I need to smile, my grand parents, Fatma and Hasan İhsan Çopur, my parents, Emine and Adnan Soğancı and my relatives Adil and Ayşe Yüceler.

ABSTRACT

PARALLEL IMPLEMENTATION OF A VQ-BASED TEXT-INDEPENDENT SPEAKER IDENTIFICATION

Automatic user identification is an indispensable part of today's computer based applications. Passwords and keys are common solutions due to their ease of implementation and low cost, but these solutions also contain the risk of being forgotten, stolen, or being used by unauthorized users, therefore security professionals are working on biometric solutions that are based on human specific characteristics. Biometric solutions include a great range from iris-scan to finger-scan, from DNA analysis to signature and keystroke scan. Voice is a popular biometric as it can be easily collected and digitalized by a microphone set or by a phone.

In this study a text-independent speaker identification system is presented. Mel-Frequency Cepstrum Coefficients are used in feature extraction, Linde-Buzo-Gray vector quantization is used in modeling these features, and measuring the similarity of models is achieved by using Euclidean distance metric. Comparing meaningful characteristics of voice samples requires a significant amount of transformations and calculations; therefore speaker recognition process results with large amount of memory usage and disk access. To share this load to a cluster system instead of using a serial machine, a parallel text-independent speaker identification system is implemented, and clear performance improvements are observed. Our parallel speaker recognition system achieves a speed up about 13.8 compared with its serial implementation in the case of using 16 processing elements to identify a corpus of 100 speakers.

ÖZET

VECTOR NİCELİKLENDİRMESİNE DAYALI METİN BAĞIMSIZ SES TANIMA SİSTEMİNİN PARALEL UYGULANMASI

Günümüzde otomatik kimlik tanıma sistemleri bilgisayar tabanlı uygulamaların vazgeçilmez bir parçasını teşkil etmektedir. Bu amaçla üretilen şifre ve anahtar gibi çözümler düşük maliyetleri ve kolay uygulanabilirlikleri ile oldukça yaygın kullanılmaktadır, ancak bu çözümler unutulma, kaybedilme ya da yetkisiz kişilerin eline geçebilme riski taşımaktadırlar, bu nedenle kişiye özel fiziksel karakteristiklerin belirlenmesi ve kullanılmasına dayalı biometrik çözümler üzerinde durulmaktadır. Biometrik çözümler iris taranmasından, parmak izi incelenmesine, DNA analizinden, imza ve yazı kontrolüne kadar oldukça geniş bir yelpazeye yayılmaktadır. Ses, bir mikrofon veya telefon aracılığı ile kolaylıkla sayısal veriye çevrilebilmesi nedeni ile oldukça popüler bir biometriktir.

Bu çalışmada konuşan kişinin söylediği kelimelerden bağımsız olarak kimliğini testbit etmeyi hedefleyen bir sistem sunulmaktadır. Sesin kişiye özel karakteristiklerinin belirlenmesinde Mel-Frekans Kepstrum sabitleri, bu karakteristiklerin modellenmesinde Linde-Buzo-Gray vektör niceliklendirmesi, iki modelin benzerliğinin değerlendirilmesinde Euclidean mesafesi kullanılmıştır. Ses örneklerinin anlamlı karakteristiklerinin karşılaştırılması, ve benzer olanlarının eşleştirilmesi önemli ölçüde dönüşümler ve karşılaştırmalar gerektirmektedir, bu nedenle oldukça fazla hafıza kullanımı ve disk erişimi söz konusudur. Tek bir bilgisayar üzerinde oluşan bu yük, paralel çalışan bir bilgisayar kümesine dağıtılarak daha hızlı sonuç üreten bir sistem oluşturulmuştur. 100 modelin 16 işlemci ile parallelizme dayalı eşleştirilmesi aynı methodu kullanan seri uygulamaya göre 13.8 kat hızlanma sağlamıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

a_k	Model parameters
A_i	Cross-sectional tube.areas
d	Time to calculate euclidean distance
C_{ij}	Covariance matrix
D_{ij}	Distance between two vectors
D_x	Mahalonobis distance
$d(u,v)$	Euclidean distance
e_n	Prediction error
G	Gain scaling factor
H	Hypothesis
k_i	Reflection coefficients
N	Size of train data
N(reject)	Condition that the utterance is rejected
m	Time required to calculate MFCC of one speech file
M	Size of test data
$mel(f)$	Mel frequency function
P	Number of processing elements
P(N|n)	Probability of correct rejection
P(N|s)	Probability of false rejection
P(S|n)	Probability of false acceptance
P(S|s)	Probability of correct acceptance
R	Autocorrelation
s	Sampling time for one speech file
S_i	Speaker i
S_n	Present output
S_{n-k}	Past outputs
S(accept)	Condition that the utterance is accepted as being of the customer
T	Threshold
T_{serial}	Serial execution time

$T_{parallel}$	Parallel execution time
T_{serial_test}	Serial testing time
T_{serial_train}	Serial training time
$T_{parallel\_train}$	Parallel training time
$T_{parallel\_test}$	Parallel testing time
$T_{communication}$	Communication time
U_n	Present output
$w(n)$	Windowing signal
v	Time required for vector quantization of one MFCC matrix
V	Number of templetes
$x(n)$	Input signal
X	Feature vector set
$y(n)$	Windowed signal

A/D	Analog to digital
DTF	Discrete Fourier Transform
DTW	Dynamic Time Warping
EER	Equal Error Rate
FFT	Fast Fourier Transform
FA	False Acceptance
FR	False Rejection
GLA	Generalized Lloyd Algorithm
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
LAR	Log Area Ratios
LBG	Linde Buzo Gray
LP	Linear Predictive
MAP	Maximum a posterior
MFCC	Mel-frequency Cepstrum Coefficients
MPI	Message Passing Interface
MSE	Mean Square Error
NN	Nearest Neighbor
PNN	Pairwise Nearest Neighbor

PVM	Parallel Virtual Machine
RLS	Randomized Local Search
SOM	Self Organizing Maps
SPMD	Single Program Multiple Data
VQ	Vector Quantization

1. INTRODUCTION

In the modern world, with the increased use of computers as vehicles of information technology, protecting sensitive/personal data from unauthorized access becomes a main concern for security professionals. As a result there is a growing need to authenticate and identify individuals automatically. Various authentication methods that have been discovered so far can be grouped in to three categories: something the user knows such as passwords, pins, and maiden names; something the user has such as keys, smart cards or tokens and who the user is which is based on biometrics.

Passwords traditionally have been the first line of defense against unauthorized access. As the need for security has increased, standards for passwords have become more restrictive. Today, a "strong" password is often required where users must mix numbers, and special characters to slow a potential hacker. Also system configurations such as password length, expiration, lockout, preventing reuse and timeout to properly enforce strong passwords makes it more difficult for users to remember passwords, leading to the user posting them in plain text files, which is very insecure. Users traditionally choose passwords that are easy to remember or that point to something in their everyday life; therefore making the password easy to remember but also easy to hack.

The brute force password guessing is the easiest attack on a password. This method keeps trying to guess a password until the correct password is guessed. Even the strongest passwords can be broken if there is enough time spent trying to break it. Some other attack methods are login spoofing, replay attacks, and by monitoring the traffic between computers. Although it has lots of disadvantages, a password-based authentication is very cheap thus causing it to be used widely.

Smart cards and tokens are other alternatives to password. They are not perfect but better than passwords, so they can be considered as a link between traditional password and biometric identification. Magnetic cards such as credit and ATM cards are one of the common implementations of token technology that contains information about the holder of the card. Tokens are also used in conjunction with the passwords and this increases the security. Smart cards provide an additional element of security with use of one-time passwords. There are three main smart token protocols that determine the level of security provided by the device, which are static password exchange, dynamic password generator and the challenge response. With static password exchange protocol the user authenticates to smart token, which in turn authenticates to computer. Static tokens work much like memory tokens. The key to these devices is the PIN, which is entered by the user. In dynamic password generator scheme a key is generated by a token at regular intervals. The user can then enter the password into the computer, or if the token has an electronic interface, the password is sent automatically to computer for authentication. Challenge-response protocol is based on a computer that generates a random set of numbers, which smart token uses to generate a response to computer that is used for authentication.

The main advantage of challenge-response and password generator schemes is one-time passwords. This technology is also a combination of two authentication methods, something the user knows and something user has. But, as with any technology, there are some disadvantages. The main disadvantage is the cost of token replacement and the cost of readers for electronic interface tokens. There is also a fair amount of administration involved in the use of smart token technologies.

Authentication methods that deal with something the user knows, or something the user has, have some disadvantages that cause vulnerability, therefore biometrics-based authentication is emerging as a reliable method that can overcome some of the limitations of traditional automatic personal identification technologies. Automated biometrics deal with physiological and/or behavioral characteristics. Some examples of biometrics used commercially are finger-scan (optical, silicon, ultrasound, touch less), facial-scan (optical and thermal), voice-scan, iris-scan, retina-scan, hand-scan, signature-scan, keystroke-scan, and palm-scan. There is also biometrics with reduced commercial viability, due to their high cost or implementation restrictions, such as DNA analysis, ear shape, odor (human scent), vein-scan (in back of hand or beneath palm), finger geometry (shape and structure of finger or fingers), nail bed identification (ridges in fingernails) and gait recognition (manner of walking). Since there is rapid progress made in electronics and in Internet commerce, along with the increased emphasis on security, there will be a growing need for secure transaction processing using biometrics technology.

1.1. Motivation

The aim of this thesis is to present a voice identification method that can be used for biometric user authentication. Voice samples can be collected by a simple microphone attached to a PC or by a phone, so voice identification is advantageous to many biometrics that require high cost equipment to capture related characteristics. Voice is a very effortless way of communication, so people who become restless using methods such as retina and iris scanning will feel in comfort.

Biometrics can be used to authenticate a person's claim to a certain identity or establish a person's identity from a large database. Verification contains large amounts of comparisons between input samples and data sets, as a result, the cost of computations and time required to do the comparisons are important concerns of implementation. An authentication system should be able to respond (accept the user or reject the user) in a reasonable amount of time. Growing data set sizes, which cannot fit even on the main memory of best computer available, forces disk based algorithms to be used, therefore increase the recognition time. Using a set of computers to do computations parallel to one another will provide main memories of all processing elements instead a disk based algorithm and will give better results compared with serial one machine applications. For this reason a parallel implementation of speaker recognition is the main motivation of this thesis.

1.2. Outline

In this thesis, we first present a general look at speaker recognition terms, methods, and related studies. The implementation is based on a text independent speaker verification system in which speech signals are modeled with mel-frequency cepstrum coefficients (MFCC). To compare input speech signal with the data set, the Linde-Buzo-Gray (LBG) vector quantization method is used, and Euclidean distance is selected for the calculation of the match score. These methods, their alternatives, and related terms are detailed in Chapter 2. In Chapter 3 parallelism and its benefits in speaker verification, and the implementations are explained. Results of several experiments are discussed in Chapter 4. Chapter 5 summarizes the work done and discusses the future work, which will improve the performance of the system.

2. SPEAKER VERIFICATION

2.1. Speech Based Applications

Speech waves are sound pressure waves that can be represented as analog signals. Analog signals must be sampled in order to obtain spectral content, to be modeled, and digitalized. Analysis is an important step in speech processing. In analysis, speech waveform is examined to extract time varying parameters, and to model the speech wave. Synthesis can be thought as an inverse function of an analysis. In Synthesis, speech wave is reproduced using the predefined model. The main goal of analysis and synthesis is to model the speech signal without losing any data.

Speech processing is a field with many applications that are based on analysis and synthesis [1]:

- *Speech Modification*: The goal is to alter a speech signal in order to have some desired property such as changing time-scale, pitch, and spectral modifications.
- *Speech Coding*: The purpose is to reduce the amount of data with out losing the quality of speech. It can be accepted as voice compression.
- *Speech Enhancement*: The goal is to improve the quality of degraded speech by reducing noise, removing unwanted convolutional channel distortion, and background talks.
- *Speech & Language Recognition*: The goal is to convert a speech file to a text file.
- *Speech Synthesis from text*: The goal is to convert a text file to a speech file.
- *Speaker Recognition*: Speaker Recognition is the use of a machine to verify a person's claimed identity from his/her voice. Speaker recognition is also a part of biometrics that is based upon voice.

Figure 2.1 shows the relationship of speech processing applications, speaker recognition, and voice biometrics. Some speaker recognition applications are voice dialing, banking by telephone, telephone shopping, database access services, information services, security control, voice mail, and remote access to computers.

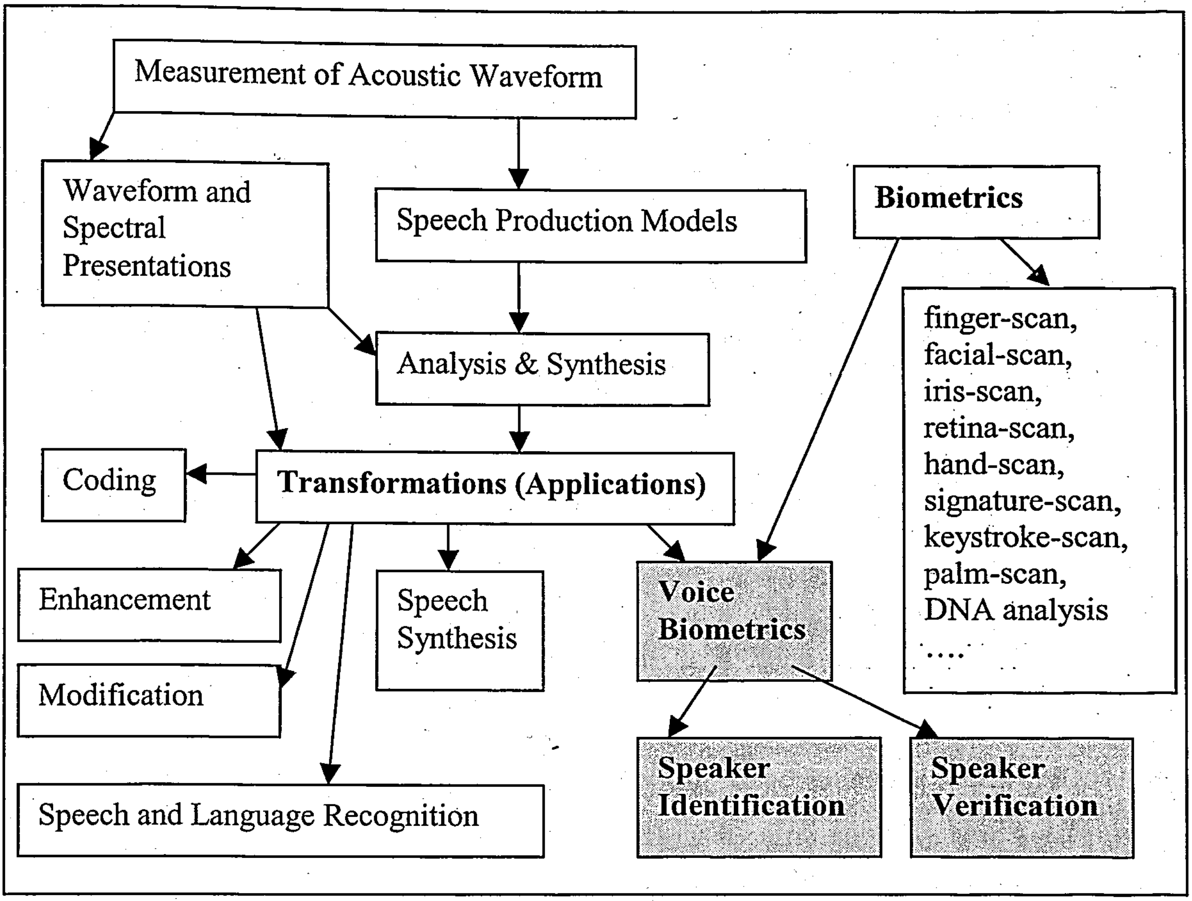


Figure 2.1. Speech processing

2.2. Speaker Recognition

Speech is a complicated signal produced as a result of several transformations occurring at semantic, linguistic, articulatory, and acoustic levels. Differences in these transformations appear as differences in acoustic properties of the speech signal and these differences produce speaker specific characteristics, which are used for recognition. Speaker specific characteristics can be grouped as physical and learned. Learned habits of speakers are speaking rate, prosodic effects and dialect. Physical differences in people that affect their speech are [2]:

- *Vocal tract shape*: Vocal tract shape is an important and popular physical distinguishing factor. While acoustic wave passes through the vocal tract, its frequency content changes with resonances in the vocal tract. These resonances are called formants, and their effects can be seen from the spectrum.

  Vocal system is also an excitation source, which is also speaker dependent. Excitation can be characterized as whispering, frication, compression, vibration, whispering and their combinations.
- *Fundamental frequency*: The frequency of oscillation, which depends on length, tension, and, mass of the vocal folds is called fundamental frequency.
- *Sub glottal resonances*: These resonances are related with the properties of the trachea.
- *Vital capacity*: It is the maximum volume of air one can blow out after maximum intake.
- *Maximum Phonation time:* Maximum duration a syllable can be sustained.
- *Glottal airflow*: The amount of air going through the vocal folds.

These speaker specific characteristics have a great potential for successful speaker recognition, but there are also some error sources such as misspoken or misread phases, extreme emotional states as stress or duress, time varying microphone placement, poor room acoustics, and using different phones in different states and sickness. Aging may also be a reason for changes in the vocal tract shape. Figure 2.2 is a representation of human vocal system.

In speaker recognition two conditions concern input utterance. One condition is that the utterance belongs to a customer (s), other is the opposite condition (n). Two decision conditions also exist: S, the condition that utterance is accepted as being that of the customer, and N the condition that the utterance is rejected. Table 2.1 gives the combination of these conditions produces four conditional probabilities [3].

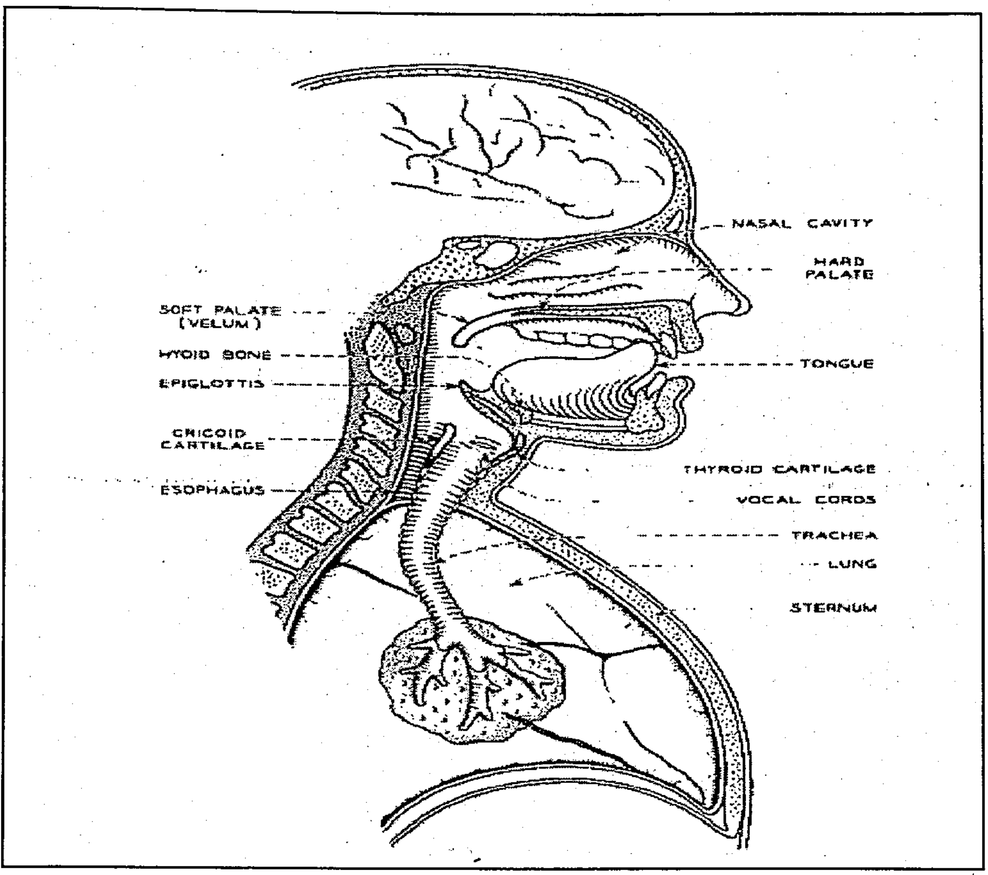


Figure 2.2. Human vocal system [2]

Table 2.1. Four conditional probabilities in speaker verification [3]

Decision Condition	Input utterance condition	
	s (customer)	n (impostor)
S(accept)	P (S | s)	P (S | n)
N(reject)	P (N | s)	P (N | n)

P (S | s) is probability of correct acceptance; P (S |n) is probability of false acceptance (FA). P (N | s) is probability of false rejection (FR) and P (N | n) is probability of correct rejection. Also, there is a relationship between decision criterion and two kinds of errors. Position a in Figure 2.3 corresponds a strict decision criterion, and position b is a lax decision criterion. In experimental tests, the threshold is usually set to a value of c in order to match up the two kinds of error rates.

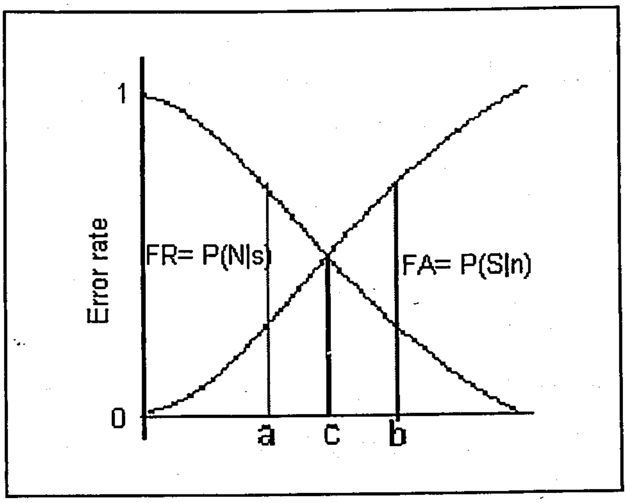


Figure 2.3. Decision criterion (threshold) [3]

Speaker Recognition can be divided into two groups according to its purpose as speaker identification and speaker verification. Also speaker recognition can be categorized as text-independent, text-dependent, and text-prompted according to what the speaker says as an input.

2.2.1. Automatic Speaker Identification

In speaker identification the goal is to recognize an unknown speaker from a set of N known speakers. The system decides who the person is, or finds that the person is unknown. There is no identity claim of the user, the system handles voice of user, and compares it with values in its predefined data set, and then computes the similarity between voices and reference models. The system returns a reference model that is most similar to user.

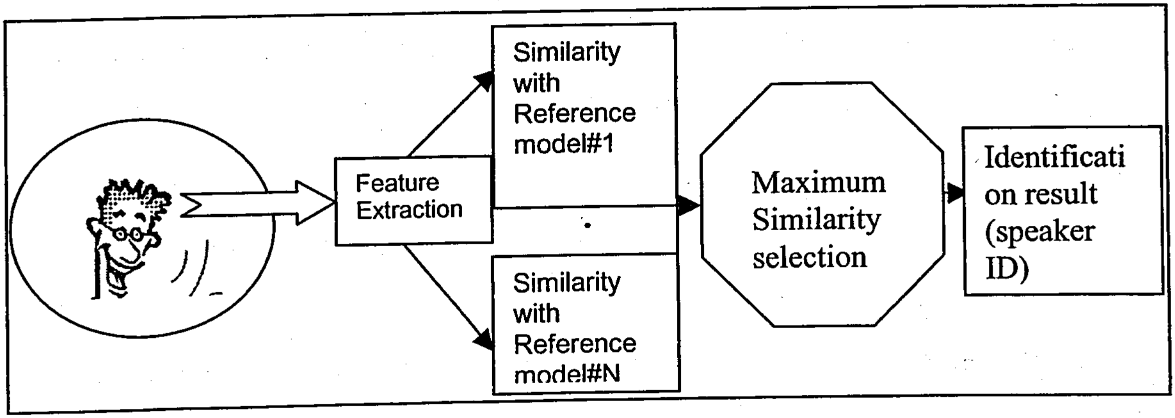

Feature Extraction
Similarity with Reference model#1
.
Similarity with Reference model#N
Maximum Similarity selection
Identificati on result (speaker ID)

Figure 2.4. Speaker identification model

There is usually a threshold value to prevent meaningless matches, if the dissimilarity degree is greater than the threshold the user will be accepted as the unknown speaker.

2.2.2. Automatic Speaker Verification

In speaker verification there is a claim of the speaker. So a verification system compares input speech with only claimed reference model. If similarity is above a threshold value, the system authenticates user. If the similarity is below the threshold value, the system rejects the user.

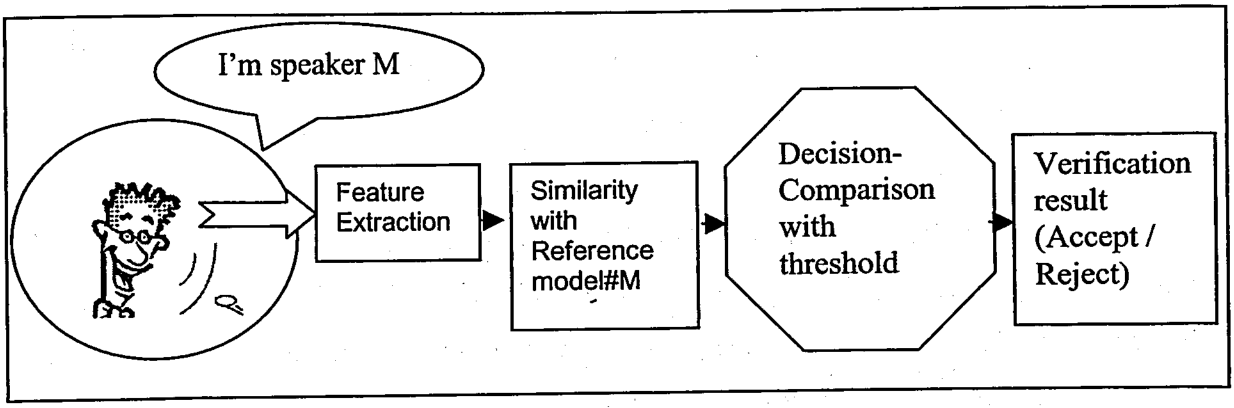

I'm speaker M
Feature Extraction
Similarity with Reference model#M
Decision-Comparison with threshold
Verification result (Accept / Reject)

Figure 2.5. Speaker verification model

2.2.3. Text-Dependent Speaker Recognition

Recognition of speaker's identity is based on his/her speaking one or more specific phrases in text-dependent speaker recognition. The speaker speaks the phrase into a microphone, and the resulting signal is analyzed by a verification system that accepts, rejects, or requests additional input before making a decision.

2.2.4. Text-Independent Speaker Recognition

Recognition of speaker's identity is based on his/her speaking some random phrases in text-independent speaker recognition. The speaker speaks the phrase into a microphone and the resulting signal is analyzed by a verification system that accepts, rejects, or requests additional input before making decision.

2.2.5. Text-Prompted Speaker Recognition

Text-dependent and text-independent systems can be easily deceived if someone plays back a recorded voice of a registered speaker saying key phases. To solve this problem, each user is prompted to say a new key sentence every time system is used and the system accepts input utterance only when it decides that it was a registered speaker who repeated the prompted sentence. Speaker specific phoneme models can be used as basic acoustic units.

2.3. Speech Processing and Feature Selection

Speech processing is extracting desired information from digitalized speech signal. Signals have to be digitized before being handled by computer systems. Microphones and telephones convert an acoustic wave to an analog signal, then the analog signal is filtered by an antialiasing filter to limit bandwidth of signal approximately Nyquist rate. Nyquist rate is half of the sampling rate. Conditioned analog signal is then sampled to form a digital signal by an analog to digital (A/D) converter with 12-16 bits resolution and 8000-20000 samples per second.

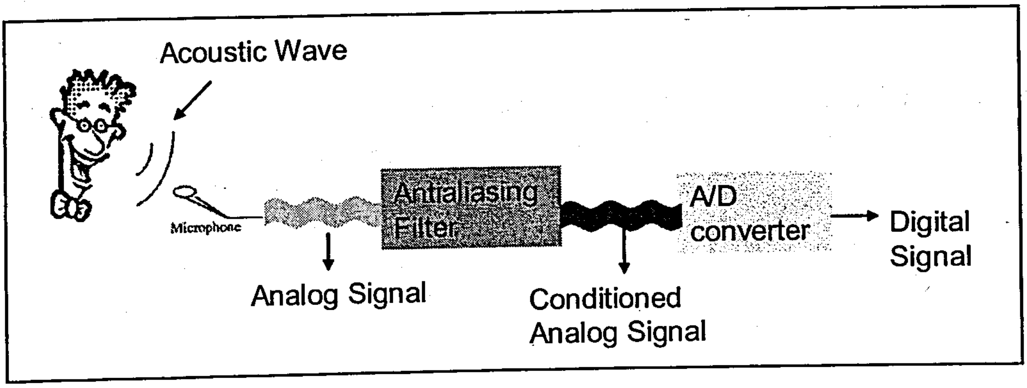


Figure 2.6. Analog to digital conversion

The first step of speech processing is modeling. There are several methods used for speech modeling. By modeling the signal, we can use some transformations to obtain speaker specific features.

Features of speech signal that minimize verification error should be selected for a successful speaker recognition system. This is called as feature selection or feature extraction. The goal of feature extraction is to find a transformation that preserves data, which will enable meaningful comparison between signals. An increased number of features will cause computational and storage overhead so feature space must be limited as much as possible. To reduce dimensionality of feature space there are several methods. Principal component analysis is a method that seeks to find a lower dimensional representation that accounts for variance of features. Factor analysis is another method that seeks to find a lower dimensional representation that accounts for correlations among features. Also, if data are linearly separable, linear transformations can be used to reduce feature space dimensions, and sometimes non-linear transformations may be useful for non-linear data.

The main target of feature extraction is to obtain a feature set that exhibits low intra-speaker variability and high inter-speaker variability. The most popular modeling methods are Linear Predictive (LP) analysis and MFCC.

2.3.1. All-pole LP Model

All-pole Linear Predictive (LP), models a signal S_n by a linear combination of its past values and a scaled present input [2].

$$S_n = -\sum_{k=1}^{P} a_k . S_{n-k} + G.U_n \tag{2.1}$$

Where S_n is present output, P is prediction order, a_k are model parameters (predictor coefficients), S_{n-k} are past outputs, G is gain scaling factor, and U_n is present input.

In speech-based applications U$_n$ is generally unknown so approximate $\hat{S}_n$ is found by ignoring U$_n$. Approximation formula is:

$$\hat{S}_n = -\sum_{k=1}^{P} a_k.S_{n-k} \tag{2.2}$$

The difference between actual and approximate values of S$_n$ is called as a prediction error. Prediction error e$_n$ can be formulated as:

$$e_n = S_n - \hat{S}_n = S_n + -\sum_{k=1}^{P} a_k.S_{n-k} \tag{2.3}$$

The mean square error (MSE):

$$MSE = \sum e_n^2 = \sum\left[S_n + -\sum_{k=1}^{P} a_k.S_{n-k}\right]^2 \tag{2.4}$$

If we take derivative of MSE for each a$_k$ and minimize it:

$$\begin{gathered}\frac{\partial MSE}{\partial a_i} = 0\ , \qquad i = 1,2,....,P \\ \sum_{k=1}^{P} a_k.\sum s_{n-k}.S_{n-i} = -\sum s_n.S_{n-i} \qquad i = 1,2,....,P\end{gathered} \tag{2.5}$$

If the summation of infinite extent, summations on s are autocorrelations at lags i-k for the left sum and at lag i for the right sum. This results in the “autocorrelation method” of LP analysis. The time-averaged estimates of the autocorrelation at lag τ can be expressed as [2]:

$$R_\tau = \sum_{i=0}^{N-1-\tau} s(i).s(i+\tau) \tag{2.6}$$

LP model parameters (a_k) are used to model speech signal by a p-dimensional a_k vector. Any signal can be defined by a linear predictor and corresponding LP error. These LP coefficients are nonlinearly transformed into feature domains such as: reflection coefficients, log area ratios, arcsine of reflection coefficients, and Linear Predictive cepstrum.

$$\begin{bmatrix} R_0 & R_1 & R_2 & \cdots & R_{p-1} \\ R_1 & R_0 & R_1 & \cdots & R_{p-2} \\ R_2 & R_1 & R_o & \cdots & R_{p-3} \\ \cdots & & & & \end{bmatrix} . \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \\ \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ \\ \end{bmatrix} \tag{2.7}$$

Reflection coefficients (k_i) can be obtained by using the following backward recursion, by using LP coefficients:

$$\alpha_j^{(p)} = \alpha_j, \quad k_i = \alpha_i^{(i)}, \qquad \alpha_j^{(i-1)} = \frac{\alpha_j^{(i)} + \alpha_i^{(i)}.\alpha_{i-j}^{(i)}}{1-k_i^2} \qquad \begin{matrix} 1 \le j \le i-1 \\ i = p, p-1, \ldots\ldots, 1 \end{matrix} \tag{2.8}$$

Log area ratio method is also used for finding reflection coefficients. Vocal tract can be modeled as a series of cylindrical acoustic tubes as demonstrated in Figure 2.7. At each junction between tubes there can be an impedance mismatch or an analogous difference, and a small portion of the wave is reflected while a rest of it transmitted. Reflection coefficients k_i gives the percentage of reflection.

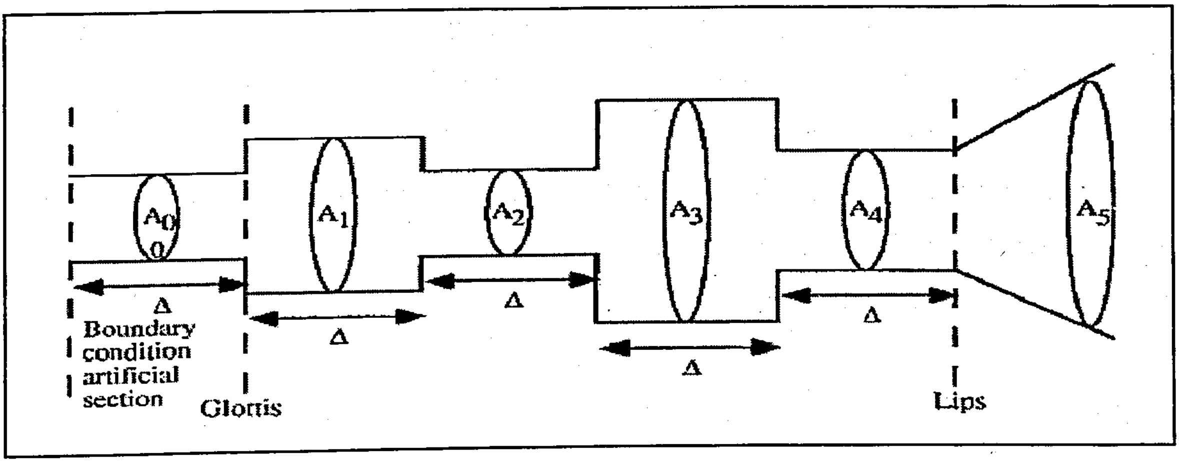


Figure 2.7. Acoustic tubes model of speech production [2]

A$_1$, A$_2$,...,A$_5$ are the cross sectional areas of the tubes, and reflection coefficients are defined as the ratios of the adjacent areas. For P cross-sectional areas:

$$\begin{gathered} A_o = 0 \\ A_{p+1} >> A_p \;\; i = 1,2,....,P \\ k_i = \frac{A_{i+1} - \mathrm{A_i}}{A_{i+1} + A_i} \end{gathered} \tag{2.9}$$

Taking the log of the area rations gives more uniform spectral sensitivity so Log Area Ratios (LAR) defined as the log of the adjacent cross-sectional areas ratios.

$$g_i = \log\left[\frac{A_{i+1}}{A_i}\right] = \log\left[\frac{1+k_i}{1-k_i}\right] = 2\tanh^{-1} k_i \qquad i = 1,2,3,...,p \tag{2.10}$$

As seen from the formula for k_i=1 there is a discontinuity in LAR, to avoid this problem using sin^{-1} of reflection coefficients are a common choice.

$$g_i' = \sin^{-1} k_i \qquad i = 1,2,3,...,p \tag{2.11}$$

2.3.2. Mel-Warped Cepstrum

Mel-warped cepstrum is a popular speech modeling method that does not require LP analysis. The first step is frame blocking. In this step, continuous speech signal is blocked into frames of N samples that intersect each other by M frames. Typically N=256 and M=100. Next step is to window each individual frame to minimize signal discontinuities at beginning and end of each frame. If we define window as *w(n)*, 0 ≤*n* ≤N -1, where *N* is the number of samples in each frame, then result of windowing is the signal:

$$y(n) = x(n).w(n), \;\; 0 \le n \le N-1 \tag{2.12}$$

Typically the Hamming window is used, which has the form:

$$w(n) = 0.54 - 0.4Cos\left(\frac{2\pi n}{N-1}\right),\ 0 \le n \le N-1 \tag{2.13}$$

The third step is Fast Fourier Transform (FFT), which is used to transfer N samples from time domain to frequency domain to obtain the spectrum of the signal. FFT is a fast algorithm to implement Discrete Fourier Transform (DFT), which is defined on the set of N samples {x$_n$}, as follow:

$$X_n = \sum_{k=0}^{N-1} x_k e^{\frac{-2k\pi jn}{N}} \qquad \text{where } j = \sqrt{-1} \tag{2.14}$$

Here, j denotes imaginary unit, X_n's are complex numbers, resulting sequence $\{X_n\}$ is interpreted as zero frequency corresponds to $n = 0$, positive frequencies $0<f<Fs/2$ correspond to values $1 \le n \le N/2 - 1$, and negative frequencies $-Fs/2<f<0$ correspond to $N/2+1 \le n \le N-1$ [1].

Human perception of speech frequency does not follow a linear scale that is measured in Hz. Mel-frequency scale is an alternative to linear scale, which is linear frequency spacing below 1000 Hz and logarithmic spacing above 1000 Hz.

To obtain mel-scaled values from a given frequency the following formula is used:

$$Mel(f) = 2595.\log_{10}\left(1 + \frac{f}{700}\right) \tag{2.15}$$

One approach to simulating subjective spectrum is using a filter bank, in a way that each desired mel-frequency component there will be one filter. Number of filters is typically chosen as 20. Figure 2.8 shows an example filter bank.

Finally log mel spectrum is converted back to time domain, this gives MFCC, which provides a good representation of speaker specific spectral properties of signal. MFCC's C_n can be calculated as;

$$C_n = \sum_{k=1}^{K} (\log S_k).\cos\left[\frac{n(k-0.5)\pi}{K}\right], \text{ n=1,2,...K} \tag{2.16}$$

where S_k is mel power spectrum coefficients.

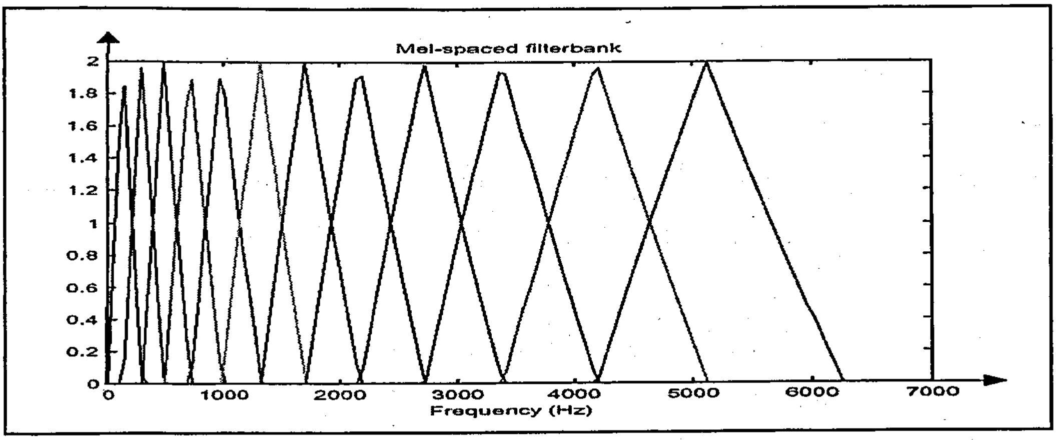


Figure 2.8. An example of mel-scaled filter bank

Set of MFCC values is called as an acoustic vector, and acoustic vectors can be used to represent and recognize voice characteristics of a speaker.

2.4. Pattern Matching

MFCC and LP are methods that are used to model a signal to form classes. To authenticate a model is constructed from incoming speech, then speech signal is compared with the model of claimed user and a match score is calculated. Match score is a measure of similarity between an input vector and a model. Two types of models are template and stochastic.

2.4.1. Template Models

In template models pattern matching is deterministic. An observation is assumed to be an imperfect replica of a template and the target is minimizing distance between model and observation. The likelihood between a model and a target is:

L= exp (-a .d), where *a* is a positive constant and *d* is distance measure. In single template models, a model of a claimed speaker could be centroid (mean) of N training vectors.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.17}$$

A match score is a distance between two patterns. Most commonly used distances are Bhattacharyya distance, Mahalanobis distance, and Euclidean distance.

Bhattacharyya distance is a measure of similarity between classes and may therefore be used to assess quality of statistics prior to classification. Distances are calculated for specified feature sets that consist of subsets of the bands in the file. For each feature set, the pair wise distance is given by:

$$D_{ij} = 0.5 \times In\left(\frac{\left|\frac{C_i + C_j}{2}\right|}{\sqrt{|C_i|}\cdot\sqrt{|C_j|}}\right) + \frac{(\mu_i - \mu_j)^T \times (\mu_i - \mu_j)}{4 \times (C_i - C_j)} \tag{2.18}$$

Where D$_{ij}$ is Bhattacharyya distance between classes i and j, C$_i$ is covariance matrix for class i, C$_j$: covariance matrix for class j, μ$_i$: mean vector for class i and μ$_j$ is mean vector for class j.

Mahalanobis distance is another way of determining the similarity between two classes. The advantage of it is taking variability of classes into account by using covariance matrix. This method is a computationally intensive method and it is formulated as:

$$D_x = (x - \mu)^T C^{-1} (x - \mu) \tag{2.19}$$

Where C is the covariance matrix, μ is the mean of the variables and x is the vector containing values.

If u=(x_1,y_1) and v= (x_2,y_2) are two points on a plane their Euclidean distance is given by : $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$. Geometrically this is the length of the segment joining u and v. To generalize this:

$$u=(x_1,x_2,x_3,\ldots\ldots,x_n), \quad v=(y_1,y_2,y_3,\ldots.,y_n)$$

$$d(u,v)=\sqrt{\left(\sum_{k=1}^{N}|x_k-y_k|^2\right)} \text{ ,where n=1,2..} \tag{2.20}$$

Where $d(u,v)$ is Euclidean distance.

2.4.1.1. Dynamic Time Warping. The Dynamic Time Warping (DTW) method that performs a piece-wise linear mapping of time between reference and input signals, and is used to compensate speaking-rate variability before match score calculation. DTW is useful when there is a variation over time. The order of DTW is O(N^2V), where N is length of sequence and V is number of templates. There are several disadvantages of DTW: first of all, it has a computation overhead with O(N^2V), which is not particularly fast, and a distance metric between frames must be defined, which may be difficult with different channels with distinct characteristics. Finally, it does not create any meaningful descriptions of data.DTW is relatively simple and has been used commerically for several industrial tasks, such as isolated spoken digit recognition.

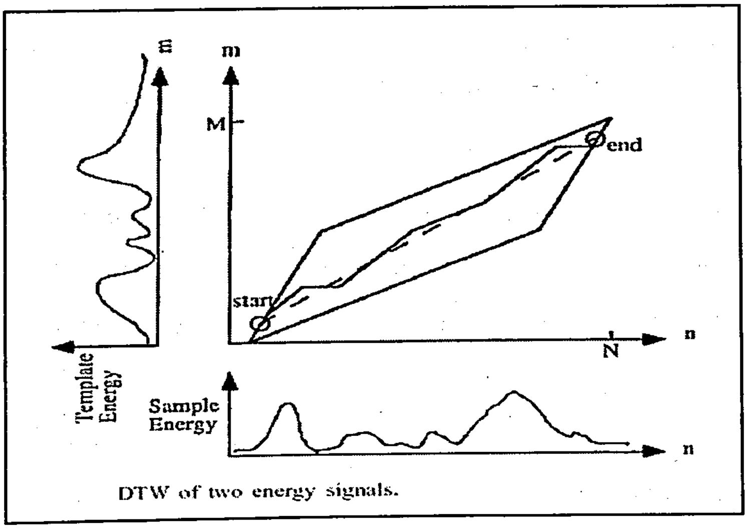


Figure 2.9. Dynamic time warping [2]

2.4.1.2. Vector Quantization. Vector Quantization (VQ) is the process of mapping vectors from a large vector space to a finite number of regions in that space such that similar vectors are grouped together. Each region in vector space is a cluster that is represented by its center. The center of a cluster is called as a centroid or a codeword. Collection of codewords is called as a codebook.

Figure 2.10 demonstrates VQ implementation with two speakers. In training phase a speaker specific vector quantization codebook is generated for each known speaker by clustering his/her acoustic vectors. Bold points in the Figure 2.10 show centroids. The minimum distance between a point and the nearest codeword is called as a vector quantization distortion.

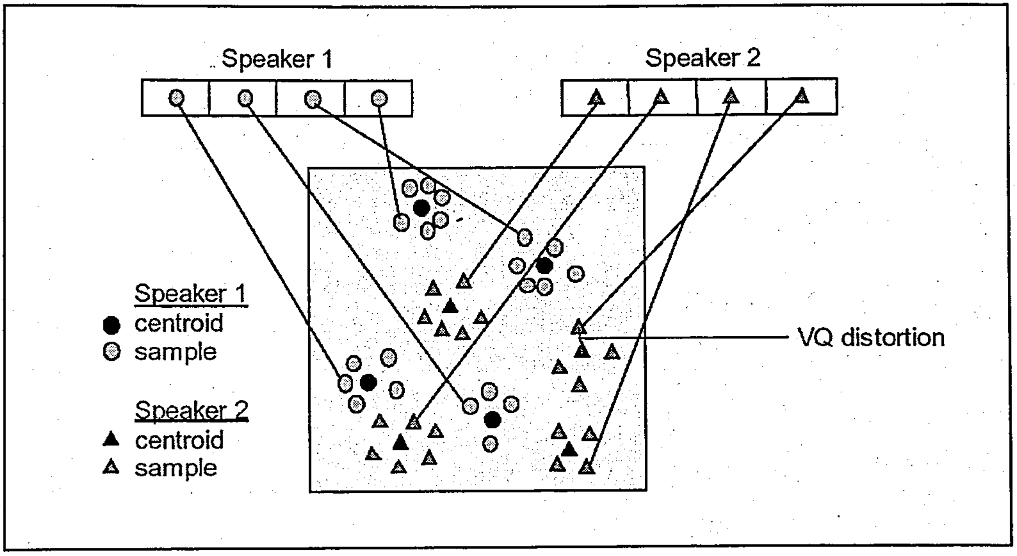


Figure 2.10. Centroid representation [4]

During the recognition phase input voice is quantized and compared with all centroids to find centroid with minimum VQ distortion. By using this method, the identity of the speaker can be understood. Identification process is composed of several steps:

- Computation of the feature vector set X. X={X_i}
- For each speaker model C_i compute the distortion between X and C_i
- Identify the index of the unknown speaker id as the smallest distortion.

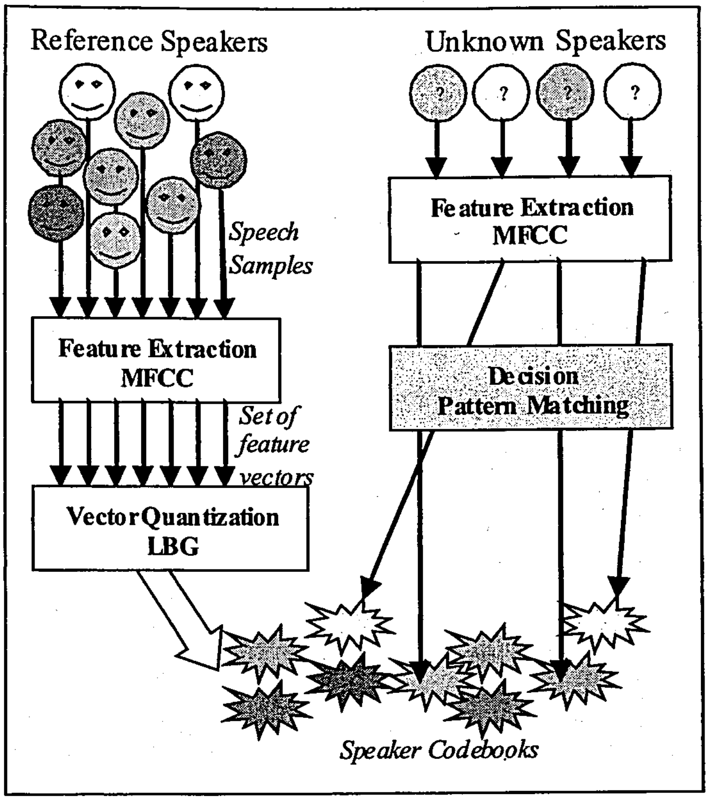


Figure 2.11. VQ based speaker identification

The simplest clustering algorithm is selecting K random feature vectors as centroids. Randomized local search (RLS) is an improvement in random selection. It starts with a random codebook and produces a predefined number of iterations. For iteration a random swap technique in which a randomly chosen code vector is replaced with another randomly chosen input vector this produces a candidate solution, then new partitions of centroids are recalculated and a new codebook is reproduced. If a candidate solution improves the previous solution it is accepted as the new solution. Self-organizing map (SOM) is another algorithm used for clustering. In SOM given a set of input patterns it learns it self how to group these patterns so that similar patterns produce similar output. SOM is a neural network based solution. Pair wise nearest neighbor (PNN) algorithm generates a codebook hierarchically. In the beginning each feature vector is accepted as a codeword and two codewords with minimum distortion are merged till the number of codewords is reduced to the desired number of centroids. Iterative splitting technique can be accepted as the opposite of PNN. It starts with an initial centroid and divides a centroid into two centroids and this splitting process continues until the centroid number increases to desired number of centroids. Generalized Lloyd algorithm, which is also called Linde-

Buzo-Gray (LBG), starts with an initial codebook and improves it iteratively until a local minimum is reached. Iteration process starts with mapping each feature vector to nearest code vector in the current book, and continues with centroid recalculation. In implementation LBG algorithm is selected for clustering.

Experimental studies are made to compare performances of these VQ algorithms. The results show that the difference between MSE is very small, but there is a marginal change in identification rates [5].

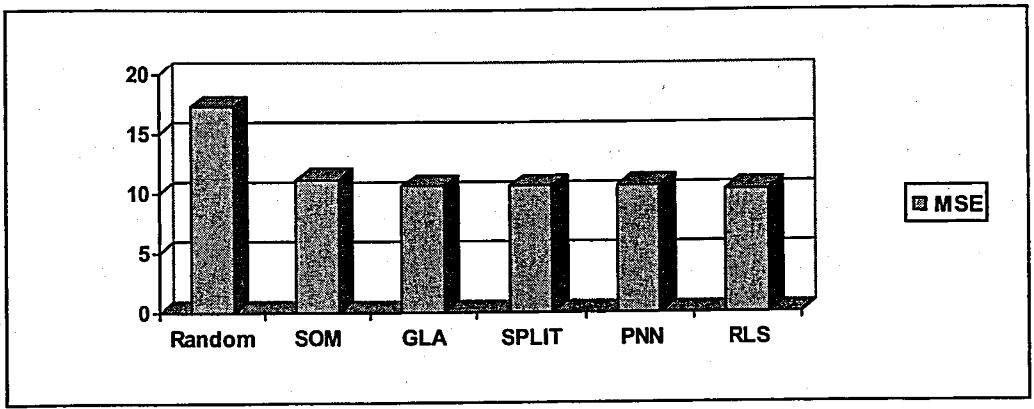


Figure 2.12. MSE values where codebook size is 64

Split algorithm is found as the fastest algorithm, and GLA algorithm is the second fastest algorithm, followed by SOM, PNN and RLS algorithms.

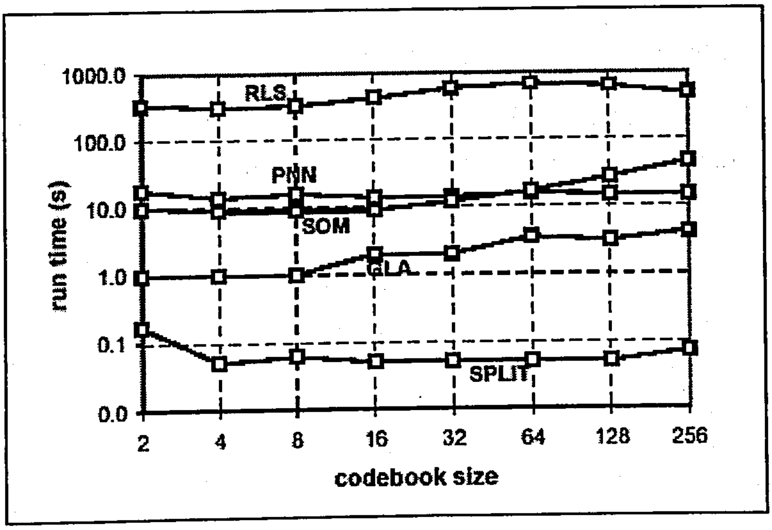


Figure 2.13. Run times of the clustering algorithms [5].

2.4.1.3. Nearest Neighbors. Nearest neighbor (NN) method combines the strength of DTW and VQ. This method does not cluster the data, instead it holds all training data in memory. NN distance is the minimum distance between a test-session frame and enrollment frames. The match score is the average of NN distances. NN is one of the most powerful pattern matching methods but it is also the most memory and computation intensive method.

2.4.2. Stochastic Models

Stochastic models such as Gaussian mixture model (GMM) and hidden Markov model (HMM) offer more flexibility and result in a more theoretically meaningful probabilistic match score. The probability of observation being generated by the claimed speaker is calculated. This probability gives match score (likelihood score). Generation of Xi by a model has the probability:

$$P(x_i \mid MODEL) = (2\pi)^{-\frac{k}{2}} \times |C|^{-\frac{1}{2}} \times \exp\left(- \frac{(x_i - \mu)^T \times (x_i - \mu)}{2 \times C} \right) \qquad (2.21)$$

where μ is mean, C is covariance, and x_i is feature vector.

2.4.2.1. Hidden Markov Model. Hidden Markov Model (HMM) is a popular stochastic method that is used to model sequences. Observations are the probabilistic functions of states. HMM is a finite-state machine, where a probability density function p (x | S_i) is associated with each S_i. The states are connected by transition network where a_{ij} are transition probabilities.

$$a_{ij} = p(S_i \mid S_j) \qquad (2.22)$$

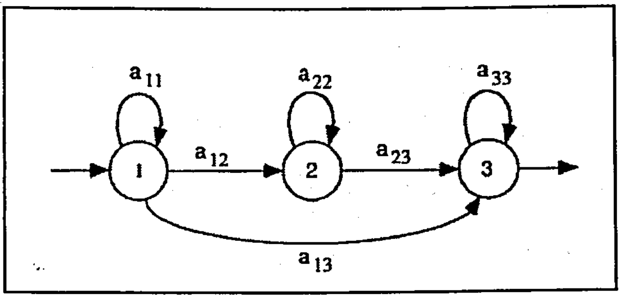


Figure 2.14. Hidden markov model [2]

The generation likelihood of L frames from model is:

$$P\,(\,x(1;L)\ \big|\ \text{model}\,)=\sum_{all\_statese\_sequences}\ \prod_{i=1}^{L} p(x_i \mid S_i).p(S_i \mid S_{i-1}) \qquad (2.23)$$

HMM performance is similar to the VQ performance.

2.4.2.2. Gaussian Mixture Model. Gaussian mixture model (GMM) can be used in both speaker verification and speaker identification systems. Speaker models can be expressed as λ_j, for j=1, 2, ..., S where S is the estimated number of target speaker models. Then for each test utterance, features at frame time n, $\underline{x}_n$ are calculated. One approach is to compute probability of the given features for each speaker model and then choose the speaker with highest probability. This approach is called as maximum a posterior (MAP) classification. To express $P(\lambda_j \mid \underline{x}_n)$ in terms of Bayes' rule:

$$P(\lambda_j \mid \underline{x}_n) = \frac{p(\underline{x}_n \mid \lambda_j)P(\lambda_j)}{P(\underline{x}_n)} \qquad (2.24)$$

Where $P(\lambda_j)$ is the priori probability of speaker λ_j, and $\underline{x}_n$ is the input feature vector. It is sufficient to maximize the quantity $p(\underline{x}_n \mid \lambda_j)P(\lambda_j)$ because P ($\underline{x}_n$) is constant. If priori probabilities are assumed equal, the problem simplifies to find $P(\lambda_j)$ that maximizes $p(\underline{x}_n \mid \lambda_j)$.

In practice there is not only one input feature vector, but a stream of input vectors generated with frame interval L. If there are M feature vectors for the utterance, $p(\{\underline{x}_0, \underline{x}_1, \ldots, \underline{x}_{M-1}\} \mid \lambda_j)$ must be maximized. In this calculation it is typically assumed that frames are independent and the likelihood for an utterance is the product of likelihoods for each frame:

$$p(\{\underline{x}_0, \underline{x}_1, \ldots., \underline{x}_{M-1}\} \mid \lambda_j) = \prod_{m=0}^{M-1} p(\underline{x}_m \mid \lambda_j) \tag{2.25}$$

By applying logarithm solution can be expressed as:

$$\hat{S} = \max_{1 \le j \le S} \sum_{m=0}^{M-1} \log\left[p(\underline{x}_m \mid \lambda_j)\right] \tag{2.26}$$

If $\hat{S}$ is the highest value for speaker i, speaker i becomes the choice for the system.

2.5. Decision

After match score computation, the system has to choose one of the two hypotheses: User is the claimed speaker or user is an impostor. Figure 2.15 show likelihood of observations from two different probability density functions according to the user is an impostor or not.

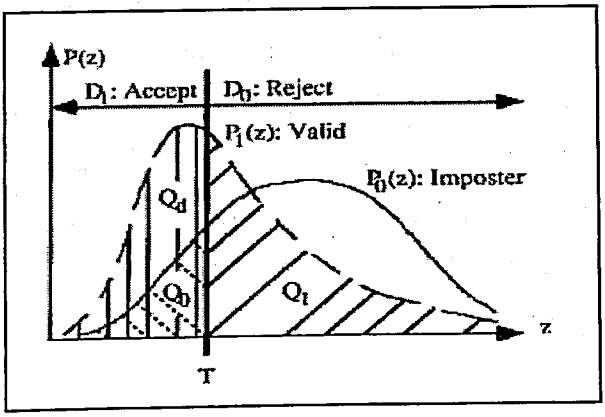


Figure 2.15. Valid and impostor densities [2]

The area under P_1 is valid user density. The area under P_0 is imposter density. The intersection areas are error sources. The area Q_0 is false acceptance of an impostor as a valid user while area Q_1 is false rejection of a valid user. T is the threshold value. As seen from the Figure 2.15 changing the threshold value will change the areas of Q_0 and Q_1 and while the area of one error type decreases the area of other error type is increasing. As a result if the flexibility of the system is increased to accept valid users who have an illness which effects the voice, crying, or under stress, false rejection will decrease, but this will be an advantage to the wolves who try to attack the system, and will increase false acceptance. Similarly if the system is too strict and tries to find almost an exact match then imposters will be avoided, but valid users may also be rejected. So the determination of the threshold value is very important.

The threshold can be determined by setting T equal to an estimate of P_1/P_0 to approximate minimum error performance, where P_0 and P_1 are the a priori probabilities that the user is an impostor and that the user is the true speaker, respectively; choosing T to satisfy a fixed FA or FR criterion or varying T to find different FA/FR ratios and choosing T to give the desired FA/FR ratio. Threshold selection can make system speaker specific, speaker adaptive, and/or risk adaptive [2].

2.6. Related Studies

There is a considerable speaker recognition activity in commerce, national laboratories, and universities. The general trend shows accuracy improvements over time with larger data sets. Table 2.2 shows a sampling of the chronological advancement in speaker verification. The column header "source" refers to a citation in references, "org" is the company or school where the work done, "features" are the signal measurements, "input" is the type of input speech, "text" indicates whether a text-dependent or text-independent mode of operation is used, "method" is the heart of the pattern matching process, "pop" is the population size, "error" is the EER for speaker verification systems "v", or the recognition error percentage for speaker identification systems [2].

Table 2.2 Selected chronology of speaker recognition process

Source	Org	Features	Method	Input	Text	Pop	Error
Atal 1974 [6]	AT&T	Cepstrum	Pattern match	Lab	Dependent	10	I:2%@0.5s V:2%@1s
Markel and Davis 1979 [7]	STI	LP	Long term statistics	Lab	Independent	17	I:2%@39s
Furui 1981 [8]	AT&T	Normalized Cepstrum	Pattern Matching	Phone	Dependent	10	V:2%@3s
Schwartz, *et al.*, 1982 [9]	BBN	LAR	Nonparametric pdf	Phone	Independent	21	I:2.5%@2s
Li and Wrench 1983 [10]	ITT	LP, Cepstrum	Pattern Match	Lab	Independent	11	I:21%@3s I:4%@10s
Dogdigton 1985 [11]	TI	Filter-bank	DTW	Lab	Dependent	200	V:0.8%@6s
Soong *et al.*, 1985 [12]	AT&T	LP	VQ(size 64) Likelihood Ratio	Phone	10 isolated digits	100	I:5%@1.5s I:1.5%@3.5s
Higgins and Wohlford 1986 [13]	ITT	Cepstrum	DTW Likelihood scoring	Lab	Independent	11	V:10%@1.5s V:4.5%@10s
Attili *et al.*, 1988 [14]	RPI	Cepstrum, LP, Autocorr.	Projected Long Tern Statistics	Lab	Dependent	90	V:1%@3s
Higgins *et al.*, 1991 [15]	ITT	LAR, LP, cepstrum	DTW Likelihood scoring	Office	Dependent	186	V: 1.7%@10s
Tishby 1991 [16]	AT&T	LP	HMM (AR mix)	Phone	10 isolated digits	100	V:2.5%@1.5s V:0.8%@3.5s
Reynolds 1995; Reynolds and Carlson 1995 [17]	MIT-LL	Mel-Cepstrum	HMM (GMM)	Office	Dependent	138	I:0.8%@10s V:0.12%@10s
Che and Lin 1995 [18]	Rutgers	Cepstrum	HMM	Office	Dependent	138	I:0.56%@2.5s I:0.14%@10s V:0.62%@2.5s
Colombi *et al.* 1996 [19]	AFIT	Ceps, Eng, ΔCeps, ΔΔCep	HMM monophone	Office	Dependent	138	I:0.22%@10s V:028%@10s
Reynolds 1996 [20]	MIT-LL	Mel-Ceps, Mel-ΔCeps	HMM (GMM)	Phone	Independent	416	V:11%16%@3s V:6%8%@10s V:3%5%@30s
Vincent Wan, William D. Campbell 1999 [21]	University of Sheffield & Motorola	LP	Support Vector Machines (SVM)	Phone (Yoho)	Independent	138	I: 4.5 % V: 0.18%0.31 %
Woohyung *et al.*, 2001 [22]	Seaul National University	LRT, LLR, Cepstrum	GMM+UBM	Phone (OGI)	Independent	91	I: 4.5 % V: 0.18%0.31 %

Table 2.3 Selected chronology of speaker recognition process (Cont.)

Source	Org	Features	Method	Input	Text	Pop	Error
Ran D.Zilca 2001 [23]	IBM	SG, DSR, MFCC	Covariance Modeling	phone (1999 NIST male segment)	Independent	230	SNST V:10-14%@15-45s DNST V:19-28%@15-45s DNDT V:47-74%15-45s
Ganchev *et al.*, 2002 [24]	University of Patras	MFCC	Probabilistic Neural Networks+GMM, VQ k-means	Polycost and SpeechDatII databases	Dependent (PIN & 10 digits)	110 400	I:5.35%@43s V:1.97%@43s I: 13.37%@43s V: 4.08%@43s
Eric Chang *et al.*, 2002 [25]	Microsoft Research Asia	MFCC	HGMM	Phone NIST 99 NIST 02	Independent	539 10	V: 12%@120s V: 4.3%@120s
Ghing Tang *et al.*, 2003 [26]	Tamkang University	LP Wavelet Packet Transform	GMM,	Phone (MAT database)	Independent	400	Identification Rate: 91.24%

Although it is not meaningful to make comparisons between text-independent and text-dependent tasks, the table provides a brief summary for the history of speaker verification.

3. IMPLEMENTATION

Implementation is based on a text independent speaker verification system in which speech signals are modeled with MFCC. To compare input speech signal with data set, the LBG VQ method is used and the match score is found by calculating Euclidean distance. After serial implementation, parallelism is achieved by MPI library calls from standard C language.

Both serial and parallel implementations contain two phases: training and testing. In training phase the data set is processed and a modeling structure based on VQ is implemented. In the testing phase, which is also named as authentication phase, incoming speakers are modeled and these models are compared with existing models in order to obtain matches between speaker models to identify the speaker's identity. In the system there are two speech files for each speaker, which are not identical. One file of each speaker is in the training set, and other file of each speaker is in the testing set. System matches two files of same speaker, which means system recognizes the speaker.

High quality speech files, which belongs to YOHO Speaker Verification corpus [2] that was collected by ITT under a US government contract is used in this study. The YOHO database was the first large-scale scientifically controlled and collected, high-quality speech database for speaker verification testing at high confidence levels. The database is in digital form. There are 106 males and 32 females joined the study using their voices. Speech signals are collected with a STU-III electret-microphone telephone handset over a three-month period in a real world office environment. There are four enrollment sessions per subject with 24 phrases per session and ten verification sessions per subject at approximately 3-day intervals with four phrases per session. The corpus contains 1,380 validated test sessions sampled with 8 kHz and 3.8 kHz, analog bandwidth. The data size is 1.2 gigabytes.

3.1. Serial Implementation

In training phase, every speech file in train set are sampled, and 320 data points per file are obtained. These values are located into a matrix and their MFCC are calculated. Then, MFCC values are passed to vector quantization function to form regions and centroids of the data.

In authentication phase, speech files in test set are read one by one, and their MFCC's are calculated similarly. Then Euclidean distance between MFCC of files and the vector quantized data are calculated, then, test and train files with minimum distances are matched. Figure 3.1 is the operational flow of our proposed speaker identification system:

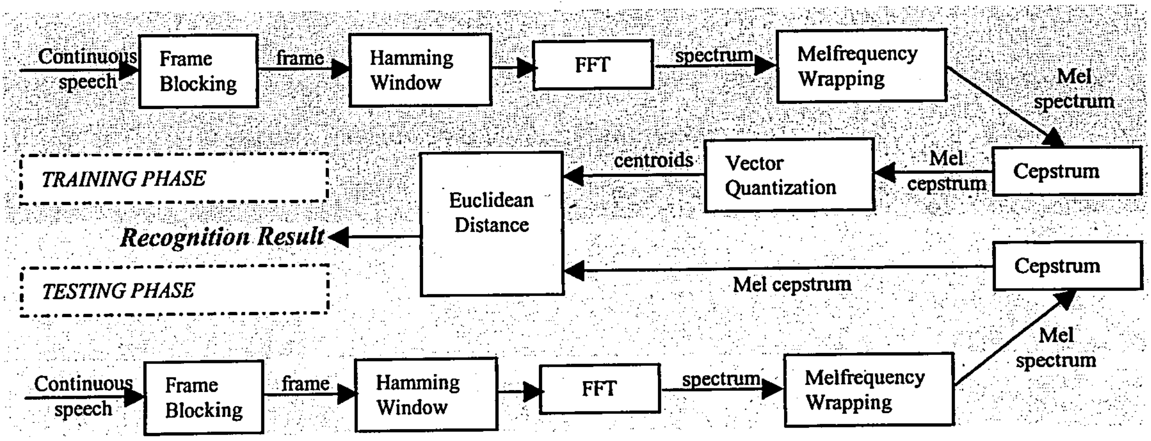


Figure 3.1. Operational flow of speaker verification system

The training files are in a folder named "enroll", and the test files are in a folder name "authenticate". Both enroll and authenticate folders contain speech files, that are named as S_1, S_2,.....,S_n in a way that the indexes give speaker id. The system compares all files in enroll folder with all files in authenticate folder, and gives files that match each other. The output "*Speaker m matches with speaker m*", means that there is a successful match between test and train data. Figure 3.2 gives the pseudo code of the serial implementation.

```
Training Part
for (int i=1; i<=N; i++){
   digitalfile=sampleFile(trainingFiles(i));
   featureVector= mfcc(digitalfile);
   VQ[i]= vqlbg(featureVector);}
Testing Part
threshold=somePredefinedValue;
for (int i=1; i<=M; i++){
   digitalfile=sampleFile(testingFiles(i));
    featureVector= mfcc(digitalFile);
    minDist= infinity;
    matchedSpeaker=0;
    for (int j=1; j<=N; j++){
    distance=euclidean(featureVector , VQ[i]);
       if (distance< minDist){
          minDist=distance;
           matchedSpeaker=j; } }
    if (minDist<threshold){
      printf("Speaker %d matches with speaker %d\n",i; matchedSpeaker);}
    else{  printf( "Speaker %d is UNKNOWN \n", i);  }
}
```

Figure 3.2. Pseudo code of serial implementation

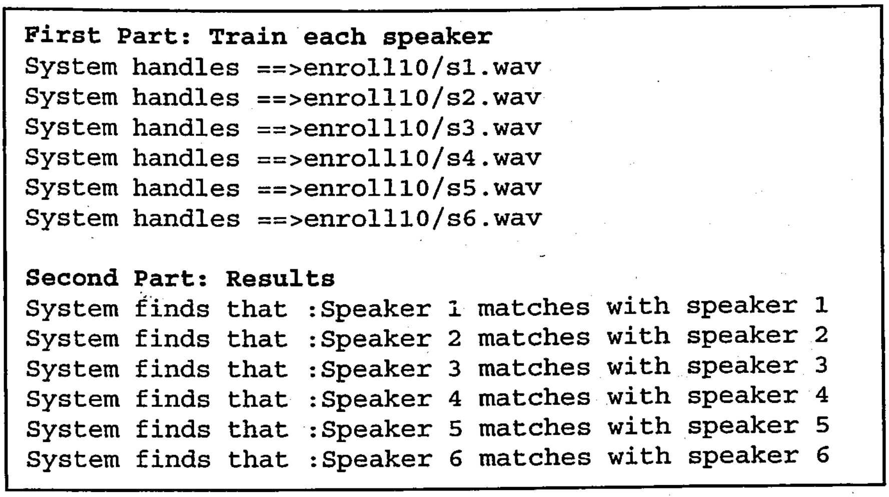
Equation 3.1 gives the execution time for serial implementation:

$$
\begin{aligned}
T_{serial} &= T_{serial\_train} + T_{serial\_test} \\
T_{serial\_train} &= N \times (s + m + v) \\
T_{serial\_test} &= M \times (s + m + N \times d) \\
T_{serial} &= M \times N \times d + N \times v + (s + m) \times (N + M)
\end{aligned}
\tag{3.1}
$$

Where; N is the size of test data, M is the size of train data, s is the time for sampling one speech file, m is the time to calculate mfcc of one file, v is the time of vector quantization of one mfcc matrix and d is the time to calculate Euclidean distance. If the train set and the test set are in the same size, Equation 3.1 will be simplified as follows:

$$T_{serial} = N^2 \times d + N \times (2s + 2m + v) \text{, where N=M} \tag{3.2}$$

The screen shot of an example execution of serial implementation for a six file small set is given in Figure 3.3.



```
First Part: Train each speaker
System handles ==>enroll10/s1.wav
System handles ==>enroll10/s2.wav
System handles ==>enroll10/s3.wav
System handles ==>enroll10/s4.wav
System handles ==>enroll10/s5.wav
System handles ==>enroll10/s6.wav

Second Part: Results
System finds that :Speaker 1 matches with speaker 1
System finds that :Speaker 2 matches with speaker 2
System finds that :Speaker 3 matches with speaker 3
System finds that :Speaker 4 matches with speaker 4
System finds that :Speaker 5 matches with speaker 5
System finds that :Speaker 6 matches with speaker 6
```

Figure 3.3. Run results for N=6, M=6

After obtaining a successful serial implementation, we try to improve its recognition rate, and reduce run time to provide a strong base for our parallel implementation. For this purpose we change some parameters to see their effects. Changing several parameters such as the VQ size, threshold and number of MFCC filter banks change both the recognition accuracy rate and the runtime, so we use the optimum values in our implementation. A detailed set of experiments on these parameters is presented at Chapter 4. Figure 3.4 gives the flowchart of serial implementation.

3.2. Parallel Implementation

Growing data set sizes, which cannot fit main memory of best computer available, and optimized serial algorithms are factors that increase popularity of parallel programming. Parallel programming methods provide ability to access greater memory and central processing unit than serial computers, so they are more suitable for large and memory intensive problems.

Success of a parallel implementation can be defined by its speed-up values. Speed-up is the ratio of execution time between the serial and parallel implementations, which depends on parts of the program that have to run serially, and the communication cost between processors during parallel computation.

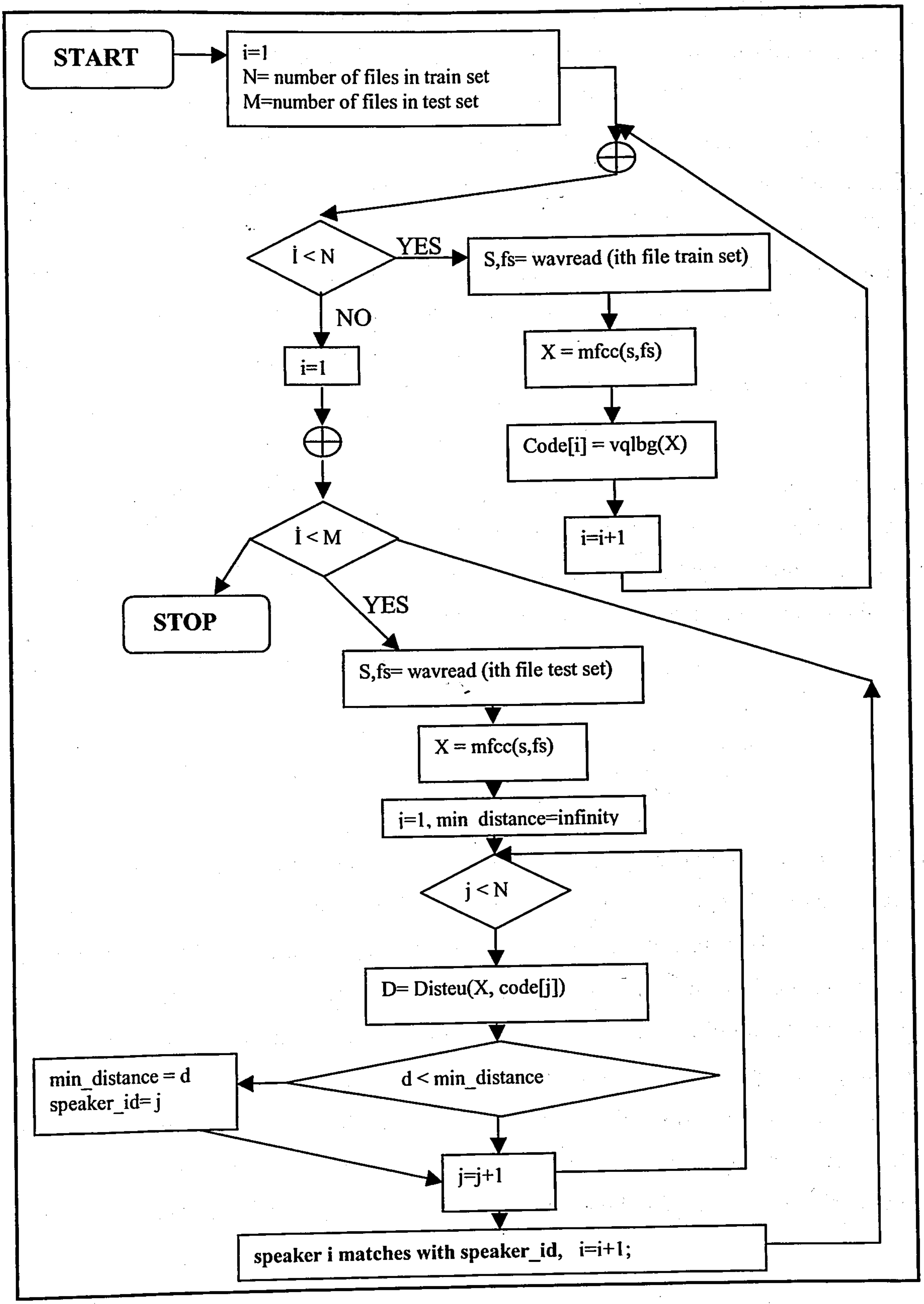

START
i=1
N= number of files in train set
M=number of files in test set
İ < N
YES
S,fs= wavread (ith file train set)
X = mfcc(s,fs)
Code[i] = vqlbg(X)
i=i+1
NO
i=1
İ < M
STOP
YES
S,fs= wavread (ith file test set)
X = mfcc(s,fs)
j=1, min distance=infinity
j < N
D= Disteu(X, code[j])
d < min_distance
min_distance = d
speaker_id= j
j=j+1
speaker i matches with speaker_id, i=i+1;

Figure 3.4. Flowchart of serial implementation

In our study first of all we want to identify the potential parallelism in speaker verification, so we try to find sections of computer code that can be executed at the same time as other sections, without changing the results generated by the program. These sections also should be the time consuming portions for obtaining good speed-up, so we look for expensive loops where iterations can be processed independently. Both training and testing parts of serial implementation handle speech files one by one and performs similar calculations for each file independently from each other so sharing the speech files between processors seem logical, thus the parallel speaker recognition algorithm presented in this work is based on domain decomposition.

We achieve parallelism in the most outer expensive loop so there is not a serial portion of the program, and time spent by a processor on serial part of a program is zero. Using Equation 3.4 we expect a linear speed-up with respect to number of processors in our case. Section 3.2.1 gives the details of MPI, which is followed by a section on Parallel Vector Quantization implementation details are given, and experimental results are presented in Chapter 4.

3.2.1. Message Passing Interface

Message Passing Interface (MPI) is the first message passing international standard where sixty people from 40 organizations were involved in its design. After two years of proposals the MPI document was produced describing its functionality. MPI is a document, which describes the interface, and there are several implementations of this standard [27]. One of the reliable implementations is the LAM-MPI that is used in our study. Message passing computation consists of a cluster of processors with local memory. Each processor executes its own program and communication is in the form of messages among processors. In the implementation, data sharing between processors is achieved by MPI calls. MPI groups are solid, efficient and deterministic, and buffer management is efficient. MPI is portable and has been a standard since the spring of 1994.

MPI consists of functions in C or subroutines in FORTRAN. These functions may be used by MPI calls that may be divided into four categories [28]:

i. Calls that are used to initialize, manage and terminate communications.

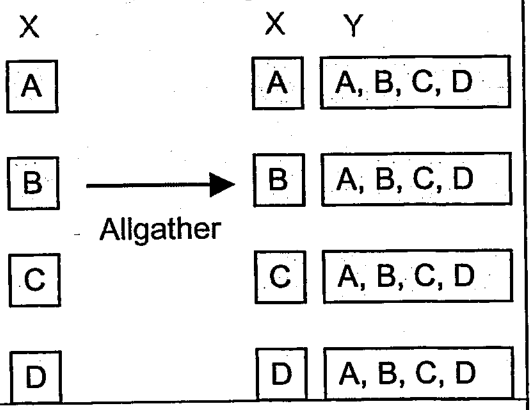
ii. Calls that are used to communicate between pairs of processors.

iii. Calls that perform communication operations among groups of processors. Groups of processors are also named communicators.

iv. Calls that are used to create arbitrary data types.

Table 3.1 gives the definitions, descriptions and example usage of the MPI functions that are used in implementation:

Table 3.1. Basic MPI functions

Definition	Example Use	Description
int **MPI_Init(** int *argc, char ***argv)	MPI_Init(&argc, &argv);	Initializes the MPI environment
int **MPI_Finalize()**	MPI_Finalize();	Does various cleanup to terminate MPI environment
int **MPI_Comm_size(** MPI_Comm comm, int *size);	MPI_Comm_size (MPI_COMM_WORLD, &numb_of_PE);	numb_of_PE variable stores the total number of processes
int **MPI_Comm_rank(** MPI_Comm comm, int *rank);	MPI_Comm_rank (MPI_COMM_WORLD, &pID);	pID stores the process id of the current process
int **MPI_Allgather(** void *sendBuf, int sendCount, MPI_Datatype SendDataType, void *recvBuf, int recvCount, MPI_Datatype recvDataType, MPI_Comm comm)	MPI_Allgather(mycode, 320*(N /numb_of_PE), MPI_DOUBLE , mycode2 , 320*(N /numb_of_PE), MPI_DOUBLE, MPI_COMM_WORLD);	Broadcast data to all the processes.  X A B C D Allgather X Y A A, B, C, D B A, B, C, D C A, B, C, D D A, B, C, D
int **MPI_Allgatherv(** void *sendBuf, int sendCount, MPI_Datatype SendDataType, void *recvBuf, int *recvCounts, int * displs MPI_Datatype recvDataType, MPI_Comm comm)	MPI_Allgatherv(mycode , sendcount , MPI_DOUBLE , mycode2 , recvcount, displs, MPI_DOUBLE, MPI_COMM_WORLD);	Broadcast data to all the processes. The size of the data, which will be broadcasted, can be different for each process in this function.

3.2.2. Parallel Vector Quantization

Data clustering is one of the fundamental techniques in scientific data analysis and data mining. It partitions a data set into groups of similar items, as measured by some distance metric. To cluster such data sets efficient parallel algorithms are called for, both to reduce computation time, and bring resources of multiple machines to bear on a given large problem in order to scale up the largest problem size one can handle [29].

Dhillon and Modha work on data clustering on a distributed memory multiprocessors. They focus on parallelizing classical k-means algorithm in order to analyze heaps of unstructured text documents. Their parallel k-means algorithm design is based on Single Program Multiple Data (SPMD) model using message passing. They observe linear speed up and excellent scale up for massive data sets and observe that k-means algorithm is inherently data parallel [30]. Patané and Russo [31] make a study to observe speed up in parallel LBG and ELBG algorithms. Their work shows that speed up increases when complexity of problem increases. They obtain a speedup about 11.24 in the best case using LBG and 6.35 in the best case using ELBG algorithm. Forman and Zhang [29] examined speed-up behaviors of k-means, k-harmonic means and Expectation Maximization algorithms and conclude that these algorithms exhibit good linear speed up.

In our implementation we used LBG vector quantization, which is also called k-means algorithm to cluster speech signals in parallel in order to classify similar speaker models.

3.2.2. Implementation Details

The parallel speaker recognition algorithm presented in this work uses domain decomposition based on file sharing, where the data is divided among the processes by considering approximately the same number of files on each processor. Source code written in Matlab is transferred to C-Mex code using Matlab compiler provided with Matlab 6.5, and parallelism is achieved by message passing interface (MPI) calls. Figure 3.5 shows the steps of our parallel implementation.

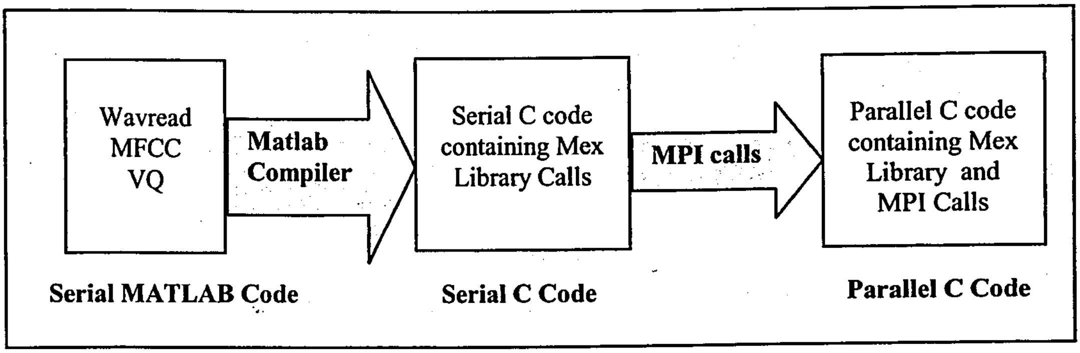

Wavread
MFCC
VQ
Matlab Compiler
Serial C code containing Mex Library Calls
MPI calls
Parallel C code containing Mex Library and MPI Calls
Serial MATLAB Code
Serial C Code
Parallel C Code

Figure 3.5. Steps of implementation

In this study speech files are shared between processes and calculations are made in parallel. Unlike serial implementation in which the algorithm runs serially in one process, in parallel implementation each process has a process id and makes calculations for a portion of the data. Each processor deals with its portion by using its processor id, and after making necessary calculations broadcast data, which will be a requirement for other processes. As a result algorithm is based on message passing, and MPI functions will be used for parallel programming. Parallel implementation starts with MPI_Init and after training and testing parts, ends with MPI_Finalize. Figure 3.6 gives the pseudo code for parallel implementation.

Running time of this implementation is the sum of running times of training and testing parts, and the communication time between processors:

$$T_{parallel} = T_{parallel\_train} + T_{parallel\_test} + T_{communication}$$

$$T_{parallel\_train} = \frac{N}{P} \times (s + m + v), \quad T_{parallel\_test} = \frac{M}{P} \times (s + m + N \times d) \qquad (3.3)$$

$$T_{parallel} = \frac{N}{P} \times (s + m + v) + \frac{M}{P} \times (s + m + N \times d) + T_{communication}$$

Where; P is the number of processing elements, N is the size of test data, M is the size of train data, s is the time for sampling one speech file, m is the time to calculate MFCC of one file, v is the time of vector quantization of one MFCC matrix and d is the time to calculate Euclidean distance. If the size of train and test sets are equal then:

$$T_{parallel} - T_{communication} = \frac{1}{P}\left(N^2 \times d + N \times (2s + 2m + v)\right) \text{ where N=M}$$

$$T_{parallel} - T_{communication} = \frac{1}{P} \times T_{serial} \text{ where N=M} \qquad (3.4)$$

$$Speedup = \frac{T_{serial}}{T_{parallel}} = \frac{T_{serial}}{\frac{1}{P} \times T_{serial} + T_{communication}}$$

Equation 3.4 gives the speed-up for our solution, if there were no communication between processors a linear speed up value P will be observed, but this is not the real case, so we expect that the computation time dominates the communication time so there will be almost linear speed up.

```
Training Part
P=MPI_Comm_size();
X=MPI_Comm_rank();
if (x < (N mod P)){
  startpos=X*((N Div P) +1)+1;
  endpos=startpos+(N div P) +1;}
else{
  startpos= (N mod P) * ((N div P))+1)+ (X-(N mod P))*(N div P)+1;
  endpos= startpos+(N div P);}
r=0;
for (int i=startpos; i<=endpos; i++){
    digitalfile=sampleFile(trainingFiles(i));
    featureVector= mfcc(digitalfile);
    VQ[r]= vqlbg(featureVector); r++;}
MPI_allgatherv(VQ);

Testing Part
P=MPI_Comm_size(); X=MPI_Comm_rank();
if (x < (M mod P)){startpos=X*((M Div P) +1)+1;
            endpos=startpos+(M div P) +1;}
else{
  startpos= (M mod P) * ((M div P))+1)+ (X-(M mod P))*(M div P)+1;
  endpos= startpos+(M div P);}
threshold=somePredefinedValue;
for (int i=startpos; i<=endpos; i++){
    digitalfile=sampleFile(testingFiles(i));
    featureVector= mfcc(digitalFile);
    minDist= infinity;  matchedSpeaker=0;
    for (int j=1; j<=N; j++){
        distance=euclidean(featureVector , VQ[i]);
            if (distance< minDist){
          minDist=distance;
          matchedSpeaker=j; } }
    if (minDist<threshold){
        printf( "Speaker %d matches with speaker %d\n", i; matchedSpeaker);}
    else{ printf( "Speaker %d is UNKNOWN \n", i); }
}
```

Figure 3.6. Pseudo code of parallel implementation

Load balancing is another important issue in parallel processing. Domain composition is based on sharing speech files and in ideal case is every processor handle equal number of files, so they will start and finish their execution almost simultaneously. This will provide good load balancing and will minimize idle time of processor thus will result with better speed-up. But in practice number of processors should not have to be a divisor of number of speakers, so some processors will handle one more speaker than the others in this case. Figure 3.7 shows an example file sharing between 3 processes.

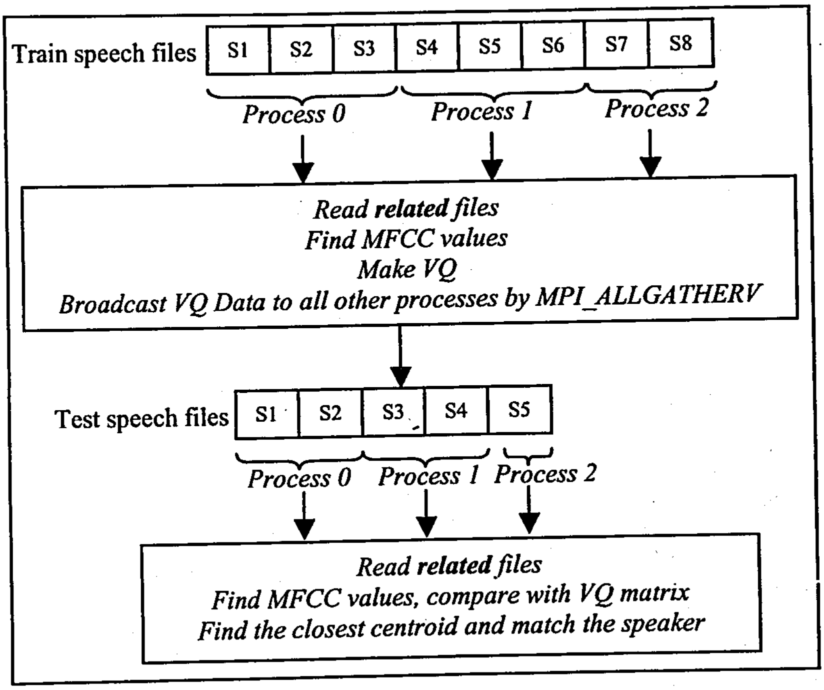
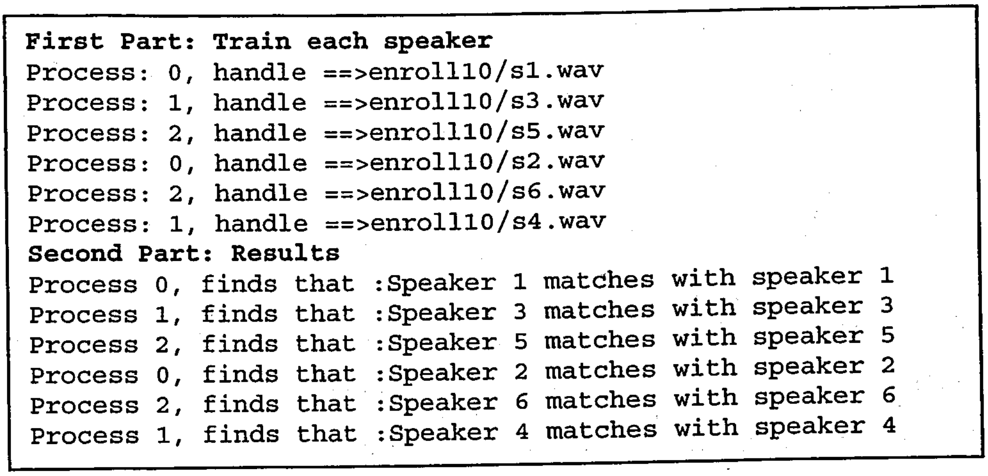


Figure 3.7. File sharing between Processes, P=3, N=8, M=5

In the example given by Figure 3.7, process 0, and process 1 handles one more file than process 2, in both training and testing phases. There is an idle time at process 2, so the execution time will remain almost the same if N=9, and M=6. If there are six files and three processes, first process handles speakers one and two, second process handles speakers three and four and the last process handles speakers five and six. Figure 3.8 shows results for such an example run.



```
First Part: Train each speaker
Process: 0, handle ==>enroll10/s1.wav
Process: 1, handle ==>enroll10/s3.wav
Process: 2, handle ==>enroll10/s5.wav
Process: 0, handle ==>enroll10/s2.wav
Process: 2, handle ==>enroll10/s6.wav
Process: 1, handle ==>enroll10/s4.wav
Second Part: Results
Process 0, finds that :Speaker 1 matches with speaker 1
Process 1, finds that :Speaker 3 matches with speaker 3
Process 2, finds that :Speaker 5 matches with speaker 5
Process 0, finds that :Speaker 2 matches with speaker 2
Process 2, finds that :Speaker 6 matches with speaker 6
Process 1, finds that :Speaker 4 matches with speaker 4
```

Figure 3.8. Run results for 6 files and 3 processors

4. EXPERIMENTS

In this section, we first present measure the effects of various parameters in our speaker identification framework, which is followed by measuring the performance of parallel implementations. The experimental study is based on YOHO corpus using the ASMA cluster system in Bogazici University, which contains 42 nodes that are connected with fast Ethernet. We used a set of 16 identical processors (p2-400 nodes with 128 MB RAM and 6 GB Hard disk) in our studies.

4.1. Experiments of Serial Implementation

In the first part, we observe the effects of the VQ size, number of filter banks and threshold value on the performance of our framework. Figure 4.1 shows effects of VQ size for 50 speakers. Increasing number of centroids increases computational complexity and results with increase in success rate and running time. In order to limit the running time, VQ size is set to 16 in our implementation.

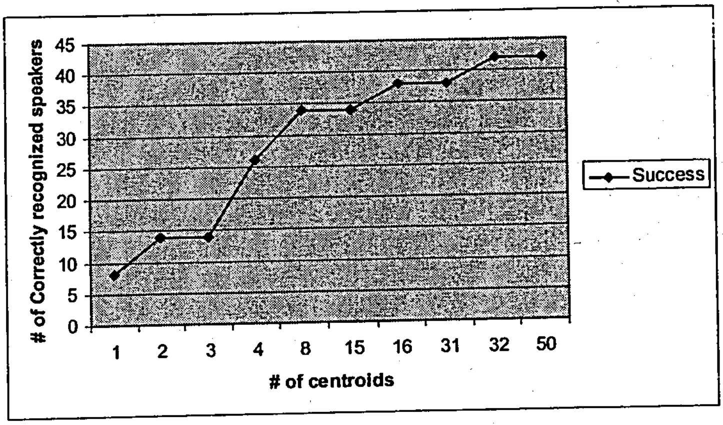


Figure 4.1. VQ size effect for N=50, M=50

Another parameter is the number of filter banks used in MFCC calculation. In our system 20 filter banks are used, but different number of filter banks are also tested to see its effect on runtime and recognition accuracy. For 50 speakers, increasing the number of filter banks up to 30, results in better success rate and less error rate, but there is an increase in running time for more than 20 filter banks. Since, there is a decrease in success rate for more than 30 filter banks, we consider 20 filter banks for the case of 50 speakers, in our experiments. Figure 4.2 gives the normalized results that are obtained by taking 5 filterbanks as a reference for the value 1.

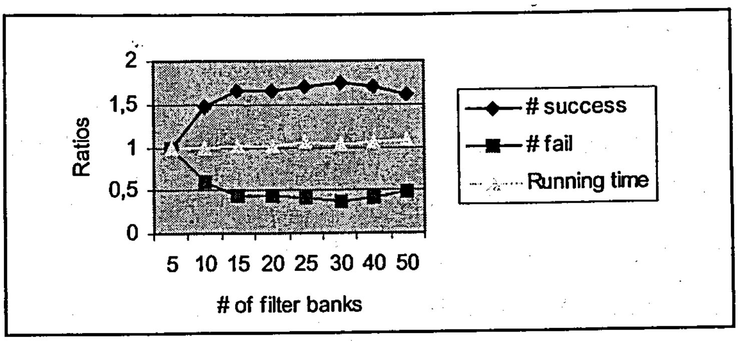


Figure 4.2. MFCC size effect for 50 speakers

In our identification system there is no threshold value, i.e., threshold is infinite; therefore, the system finds the most similar speaker for the given input. When we set threshold value, we observe false rejection. Figure 4.3 gives the number of successes, number of false acceptance and number of false rejections, for the case of 50 speakers by considering different threshold values.

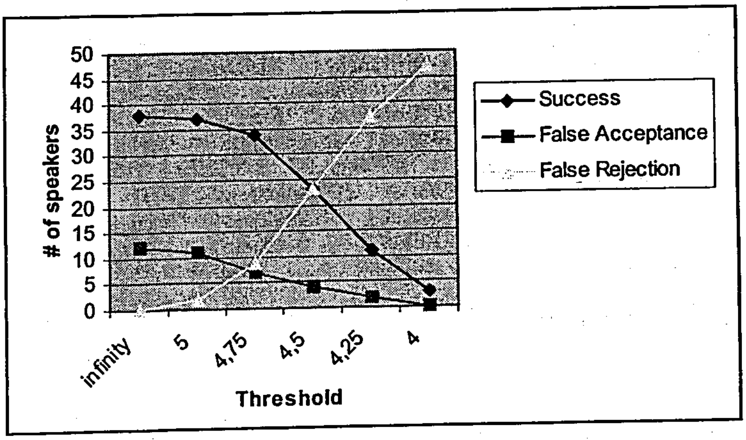


Figure 4.3. Effect of threshold value for 50 speakers

Setting threshold value reduces both false acceptance and success rate, but increases false rejection. The decrease in false acceptance is higher then the decrease in success for threshold 4.75 so it seems a suitable threshold value. Using a threshold that produces equal false acceptance and false rejection errors is called equal error rate. Results also show that threshold value 4.75 produces equal error.

4.2. Experiments of Parallel Implementation

Finally several experiments are performed on the cluster system to see effect of domain decomposition on running time. For parallel implementation, using one processor causes a slight increase in runtime compared with serial implementation, because of additional calculations for sharing data among processors. The experiments on changing population and cluster sizes result in significant decrease for running time.

Figure 4.4 shows running time cluster size relations in terms of 1000 clocks, while Figure 4.5 shows the speed up of the tests

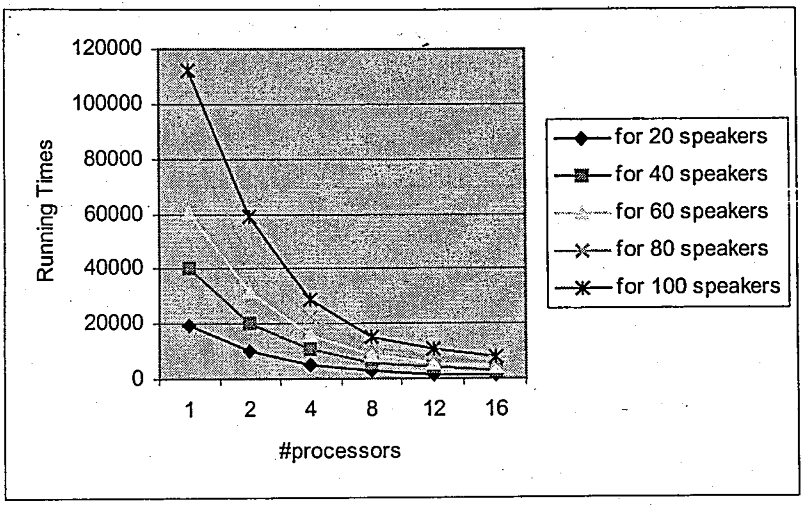


Figure 4.4. Runtime results for parallel implementation

Load balancing is based on distribution of speaker files, and the number of speakers per processors in our system and it affects the running time severely. If there are equal number of files for each processor then idle time is minimized and speed up increases, for example in 8 processors system, test with 40 and 80 speakers show better speed up than tests with 20,60 and 100 speakers as seen in Figure 4.5.

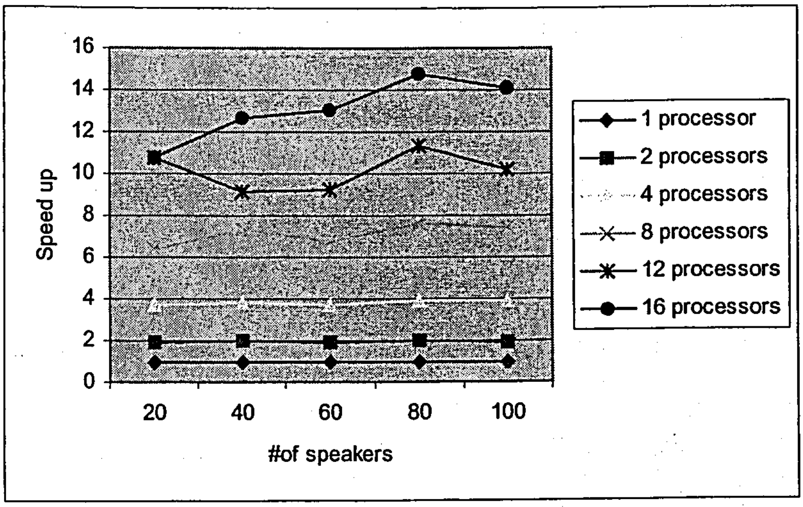


Figure 4.5. Speedup results for parallel implementation

Also reducing the maximum number of speakers per processor increases speed up, For 20 speakers using 16 processors does not improve performance compared with using 12 processors, because in both cases maximum number of speakers per processor is two. Experiments show that parallel implementation results with nearly linear speed-up in large data sets, and good scale up.

5. CONCLUSIONS

Voice is a popular biometrics that is used in identification purposes, as it is a natural way of communication and easily collected via a microphone or a phone.

In this thesis, a text independent speaker identification system based on MFCC and LBG VQ is presented by considering its parallel and serial implementations. Experiments are made with a population of 100 people that consists of 28 females and 72 males. The length of speech segments changes between 1-5 seconds and identification error is about 25% with phone data from YOHO database. In our tests, parallel implementation outperforms almost linear speed-up and good scale-up for a corpus of 100 speakers, so problem is considered suitable for parallelism. As the network cost increases severely using huge numbers of speakers, in this case train step should be divided into smaller sets, which will be trained step by step.

There are lots alternatives to our speaker verification algorithm as there are various methods that can be used in speaker modeling and pattern matching steps, so our next step is to search alternative serial implementations that may result with better recognition rates and provide their parallel implementations, as well. Also combining other biometrics such as face recognition with voice identification to strengthen recognition rates may be a starting point for human-like recognition systems.

REFERENCES

1. Quatieri, T. F., *Discrete-Time Speech Signal Processing: Principles and Practice*, Prentice Hall, 2001.

2. Campbell, J. P., "Speaker Recognition: A Tutorial", *Proceedings of the IEEE*, Vol. 85, No. 9, pp. 1437-1461, September 1997.

3. Furui, S., *Digital Speech Processing, Synthesis and Recognition*, Marcel Dekker, 2001.

4. Minh, N. D., *An Automatic Speaker Recognition System*, http://lcavwww.epfl.ch/~minhdo/asr_project/, 2004.

5. Kinnunen, T., T. Kilpeläinen, and P. Fränti: "Comparison of Clustering Algorithms in Speaker Identification", *Proceedings of the IASTED Int. Conf. Signal Processing and Communications*, Vol. SPC 2000, No. 1, pp. 222-227, 2000.

6. Atal, B. S., "Effectiveness of Linear Prediction Characteristics of the Speech Wave for Automatic Speaker Identification and Verification", *J. Acoust. Soc. Amer.*, Vol. 55, No. 6, pp. 1304–1312, 1974.

7. Markel, J. D., and S. B. Davis, "Text-Independent Speaker Recognition from a Large Linguistically Unconstrained Time-Spaced Data Base", *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-27, No. 1, pp. 74–82, 1979.

8. Furui, S., "Cepstral Analysis Technique for Automatic Speaker Verification", *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-29, pp. 254–272, 1981.

9. Schwartz, R., S. Roucos, and M. Berouti, "The Application of Probability Density Estimation to Text Independent Speaker Identification", *Proceedings of the Int. Conf. Acoustics, Speech, and Signal Processing*, Vol. ICASSP-82, pp. 1649–1652, 1982.

10. Li, K. P., and E. H. Wrench, "Text-Independent Speaker Recognition with Short Utterances", *Proceedings of the IEEE Int. Conf.Acoustics, Speech, and Signal Processing*, Vol. ICASSP, No. 2, pp. 555–558, 1983.

11. Doddington, G. R., "Speaker Recognition-Identifying People by Their Voices", *Proceedings of the IEEE*, Vol. 73, pp. 1651–1664, November 1985.

12. Soong, F. K., A. E. Rosenberg, L. R. Rabiner, and B. H. Juang, "A Vector Quantization Approach to Speaker Recognition", in *Proceedings of the Int. Conf. Acoustics, Speech, and Signal Processing*, , pp. 387–390, 1985.

13. Higgins, A. L., and R. E. Wohlford, "A New Method of Text-independent Speaker Recognition", *Proceedings of the IEEE Int. Conf. Acoustics, Speech, and Signal Processing,* pp. 869–872, 1986.

14. Attili, J., M. Savic, and J. Campbell, "A TMS32020-Based Real Time, Text-Independent, Automatic Speaker Verification System", *in Proceedings of the IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pp. 599–602, 1988.

15. Higgins, A., L. Bahler, and J. Porter, "Speaker Verification Using Randomized Phrase Prompting", *Digital Signal Processing*, Vol. 1, No. 2, pp. 89–106, 1991.

16. Tishby, N. Z., "On the Application of Mixture AR Hidden Markov Models to Text Independent Speaker Recognition", *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. 39, No. 3, pp. 563–570,1991.

17. Reynolds, D., and R. Rose, "Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models", *IEEE Trans. Speech Audio Processing*, Vol. 3, No. 1, pp. 72–83, 1995.

18. Che, C., and Q. Lin, "Speaker Recognition Using HMM with Experiments on the YOHO Database", *in Proceedings of Eurospeech*, pp. 625–628, 1995.

19. Colombi, J., D. Ruck, S. Rogers, M. Oxley, and T. Anderson,"Cohort Selection and Word Grammer Effects for Speaker Recognition", *in Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pp. 85–88, 1996.

20. Reynolds, D., "M.I.T. Lincoln Laboratory Site Presentation", http://citeseer.ist.psu.edu/murthy97robust.html, 2004.

21. Wan, V., and W. M. Campbell, "Support Vector Machines for Speaker Verification and Identification", http://www.dcs.shef.ac.uk/~vinny/docs/ pdf/svmsvm.nnsp2000.pdf , 1999.

22. Lim, W., and N.S. Kim, "Bayesian Approach to Text-Independent Speaker Verification", School of Electrical Engineering Institute of New Media and Communications Seoul National University, Seoul, Korea, 2001.

23. Zilca, R.D.,"Text-independent Speaker Verification Using Covariance Modeling", *IEEE Signal Processing Letters*, Vol.8, No.4, pp.753-756, 2001.

24. Ganchev, T., N. Fakotakis, and G. Kokkinakis, "Speaker Verification System Based on Probabilistic Neural Networks", http://slt.wcl.ee.upatras.gr/papers/ganchev3.pdf, 2003.

25. Chang, E.,"Hierachical Gaussian Mixture Model for Speaker Verification", *Proceedings of International Conference on Spoken Language Processing*, 2002.

26. PACS Training Group, "Introduction to MPI", http://www2.csit.fsu.edu/~burkardt/pdf/mpi_course.pdf, 2004.

27. Hsieh, C.T., E. Lai, and Y.C. Wang, "Robust Speaker Identification System Based on Wavelet Transform and Gaussian Mixture Model", *Journal of Information Science and Engineering* No. 19, pp. 267-282, 2003.

28. Grama, A., V. Kumar, A. Gupta, and G. Karypis, *An Introduction to Parallel Computing: Design and Analysis of Algorithms*, Pearson Addison Wesley, 2003.

29. Forman, G., and B.Zhang,"Linear Speed-up for a Parallel Non-approximate Recasting of Center-Based Clustering Algorithms, Including K-Means, K-Harmonic Means, and EM", http://citeseer.ist.psu.edu/389918.html, 2004.

30. Dhillon, I., D.Modha "A Data-Clustering Algorithm on Distributed Memory Multiprocessors", http://citeseer.ist.psu.edu/dhillon00dataclustering.html, 2000.

31. Patané, Russo, "The Enhanced LBG Algorithm". *Neural Networks*, Vol. 14 No. 9, pp.1219-1237, 2001.