**Abstract**

Speaker verification systems solve the problem of verifying whether a given utterance comes from a claimed speaker. This problem is important because an accurate speaker verification system can be applied to many security systems. Comparing to other biometric methods like fingerprint or face recognition, speaker verification systems do not require expensive specialized equipments and are effective especially for remote identity verification. Previously, Renoylds et al. have proposed a speaker verification system using Gaussian mixture model [8], but their system is incomplete because their system needs a set of background speaker models, which are constructed using a large speech database of a variety of speakers. It may not be feasible to obtain such a database in the real world. In this thesis, I propose a new solution called OSCILLO, for speaker verification. By applying tolerance interval technique in statistics, OSCILLO can verify a speaker's ID without background speaker models. This greatly reduces the size of the whole system and the time for both training and testing. We compare OSCILLO and Reynolds' method using three standard speech databases: TCC-300, TIMIT and NIST. The experimental results show that OSCILLO performs well for all databases.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Speaker verification systems solve the problem of verifying whether a given utterance comes from a claimed speaker. An accurate speaker verification system can be applied to many security systems as well as promote the accuracy of speech recognition systems.

Speaker verification is one of many applications that involve *open-set classification*. In close-set classification, a system is trained to classify an object into a finite set of pre-defined classes. During the training, the system gets to see examples from all classes. By contrast, in open-set classification, a system is required to recognize an object from a class that is unknown to the system at the training time. This is particularly difficult for classifiers that generate separation boundaries between classes, such as decision trees,

Bayesian classifiers, multi-layered neural networks, support vector machines, and many other well-known classification approaches.

Since a speaker verification system must accurately reject a speaker unknown to the system at the training time, a complete speaker verification must deal with open-set classification problem. This thesis presents a new approach to open-set classification for speaker verification problem.

## 1.2 Speaker Verification

### 1.2.1 Categories of Speaker Recognition Problems

*Speaker recognition* is the process of automatically recognizing who is speaking on the basis of information obtained from speech waves. This technique will make it possible to verify the identity of persons accessing the system, that is, implementing access control by voice, in various applications.

Lawrence Kersta made the first major step from speaker recognition when he developed spectrographic voice verification at Bell Labs in the early 1960s. He introduced the term *voiceprint* for a spectrogram, which was generated by a complex electro-mechanical device, and his verification algorithm was based upon visual comparison of these voiceprints [5]. Although the term voiceprint is a misnomer [1] and visual voiceprint comparison cannot cope with the intrinsic physical and linguistic variation in speech, the work by Kersta paved the way for automatic speaker verification.

Speaker recognition can be categorized into *text-dependent* and *text-independent* methods. The former requires the speaker to provide utterances of the same text for both training and recognition, while the latter do not rely on a specific text being spoken.

Speaker recognition can also be categorized into *speaker identification* or *speaker verification*. Speaker identification is the process of determining from which of the registered speakers a given utterance comes. Speaker verification is the process of accepting or rejecting the identity claim of a speaker. Most of the applications in which voice is used as a key to confirm the identity claim of a speaker are considered to be speaker verification.

Speaker identification can then be categorized into *open-set* or *close-set*. Open-set identification means the system has to identify the data from classes that were not part of the training set data (the close set). The problem of open-set speaker identification is the same as speaker verification.

In this thesis, we focus on the development of a text-independent speaker verification system.

### 1.2.2 Architecture of Speaker Verification Systems

Generally speaking, the architecture of a speaker verification system can be separated into two parts [2]. First, given a claimed speaker, a training system will take the speakers' speech samples as the input to construct a reference

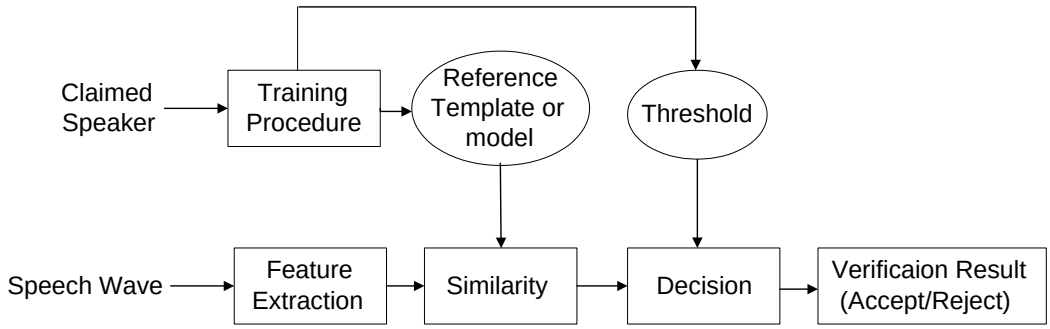


Figure 1.1: Architecture of a speaker verification system

template or model. Second, an identity claim is made by an unknown speaker and an utterance of the unknown speaker is compared with the model for the speaker whose identity is claimed. If the match is above a certain threshold, the identity is verified. The architecture is given in Figure 1.1.

### 1.2.3 Previous Work

In this subsection, we will briefly introduce some methods that have been implemented as speaker verification systems, including long-term statistics-based methods [2], vector quantization-based methods [2] and Gaussian Mixture Model based methods [10].

Long-term methods sample statistics of various spectral features, such as the mean and variance of spectral features over a series of utterances. However, long-term spectral averages are sensitive to the channel effect because these statistics are extreme condensations of the spectral characteristics of a speaker's utterances.

VQ(vector quantization) methods use codebooks which consist of a small number of representative feature vectors that are used as an efficient means of characterizing speaker-specific features. A speaker-specific codebook is generated by clustering the training feature vectors of each speaker.

GMM (Gaussian mixture model) methods use a number of Gassian models to represent a speaker's acoustic classes. The models are combined by computing the weighted sum of their outputs. Each Gaussian model and its weight are estimated by the EM algorithm [9].

These methods have been applied to the text-independent speaker verification problems. Applying the same feature extraction process, these methods have been compared empirically in terms of their verification accuracies. The results show that GMM is the most effective method for both speaker verification and speaker identification. We will describe this approach in details in Chapter 2.

## 1.3 Organization

This thesis is organized as follows. In Chapter 2, we will describe the Gaussian Mixture Model in details and a general approach to speaker verification system based on GMM. Chapter 3 presents our new approach, called OSCILLO, which is also based on GMM but applies the tolerance interval technique in statistics to allow for open-set classification. Chapter 4 com-

pares the general GMM based speaker verification system and OSCILLO using a variety of benchmark speech databases. In Chapter 5, we derive the applications of speaker verification systems and some remaining issues if we want to apply OSCILLO in practical applications. Finally, we summarize the conclusions of this thesis in the last chapter.

# Chapter 2

# Gaussian Mixture Model

In this chapter, we introduce the Gaussian Mixture Model (GMM) [9]. Mixture Models are a type of density model which comprises of a number of component functions, usually Gaussian. These component functions are combined to provide a multimodal density. This is a method which has been proved to be the most successful approach for the close-set, text-independent speaker identification system. Section 2.1 describes the mixture model. Section 2.2 explains why this method works for speaker identification and verification problem. At last, the algorithms that estimate the parameters of the model from a training database is described in Section 2.3 and 2.4. Section 2.5 reviews the general approach proposed by Reynolds [10], a GMM-based speaker verification system that requires to build background speaker models in order to reject an unknown speaker.

## 2.1 Model Description

A Gaussian mixture density of a feature vector $\vec{x}$, given the parameter vector $\lambda$ is a weighted sum of $M$ component densities, and is given by the equation:

$$p(\vec{x} \mid \lambda) = \sum_{i=1}^{M} p_i b_i(\vec{x}), \tag{2.1}$$

where $\vec{x}$ is a $D$-dimensional random vector, $b_i(\vec{x}), i = 1, \ldots, M$ are the component densities and $p_i, i = 1, \ldots, M$, are the mixture weights. Each component density is a $D$-variate Gaussian function of the form:

$$b_i(x) = \frac{1}{(2\pi)^{D/2} |\sum_i|^{1/2}} \exp\{-\frac{1}{2}(\vec{x} - \vec{\mu_i})' \Sigma_i^{-1} (\vec{x} - \vec{\mu_i})\} \tag{2.2}$$

with mean vector $\vec{\mu_i}$ covariance matrix $\sum_i$. The mixture weights satisfy the constraint that $\sum_{i=1}^{M} p_i = 1$. The complete Gaussian mixture density is parameterized by the mean vectors, covariance matrices and mixture weights from all component densities. These parameters are collectively represented by the notation:

$$\lambda = \{p_i, \vec{\mu_i}, \Sigma_i\}. \tag{2.3}$$

## 2.2 Why Gaussian Mixture Model

There are two principal advantages for applying Gaussian mixture densities as a representation of speaker identity. The first is the intuitive notion that the individual component densities of a multi-model density may model

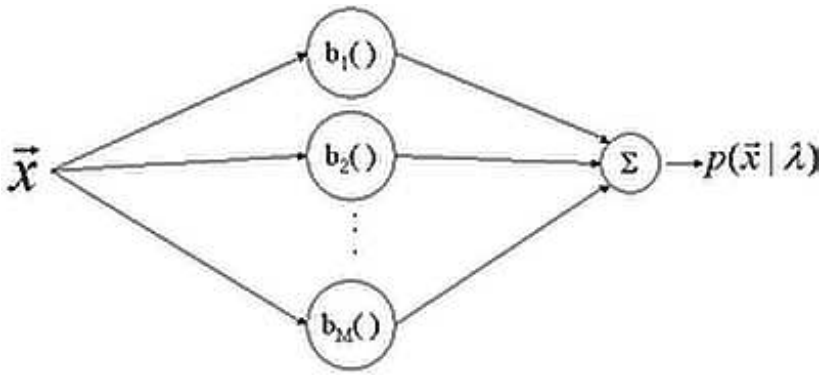


Figure 2.1: Gaussian mixture model

some underlying set of acoustic classes. These acoustic classes reflect some general speaker-dependent vocal tract configurations that are useful for characterizing speaker identity. The $i$th acoustic class can be represented by $i$th Gaussian model with mean $\vec{\mu_i}$ and covariance matrix $\sum_i$.

The second advantage of using Gaussian mixture densities for speaker identification is the empirical observation that a linear combination of Gaussian basis functions is capable of representing a large class of sample distributions. One of the powerful attributes of GMM is its ability to form smooth approximations to arbitrarily-shaped densities. In Figure 2.2, we compare the densities obtained using a unimodal Gaussian model, and a GMM density curve. GMM not only provides a smooth overall distribution fit, its components also clearly detail the multi-modal nature of the density.

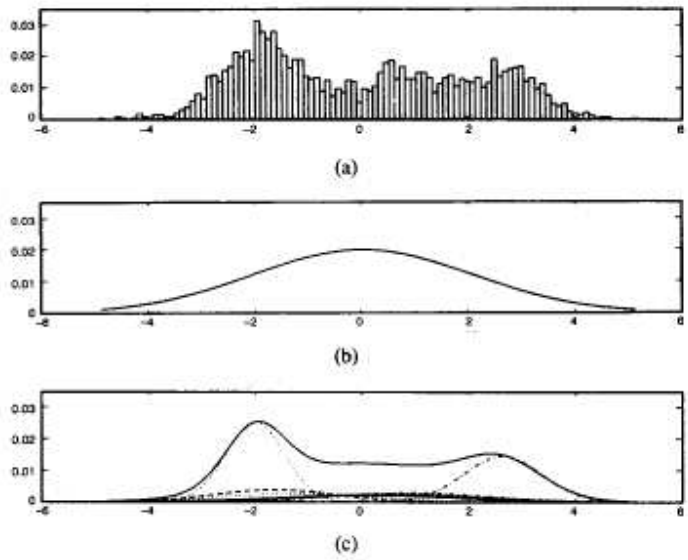


Figure 2.2: Comparison of distribution modelling (a)Histogram of a single cepstral coefficient from a 25 second utterance by a male speaker; (b)maximum likelihood unimodal Gaussuan model; (c)GMM and its 10 underlying component densities [9].

## 2.3 Estimating GMM Parameters

Given training speech form a speaker, the goal of speaker model training is to estimate the parameter vector $\lambda$ for GMM. There are many techniques available for estimating the parameters of a GMM. The most popular and will-established method is by maximum likelihood (ML) estimation.

The aim of ML estimation is to find the model parameters which maximize the likelihood of the training data. For a sequence of $T$ training vectors

$X = \{\vec{x_1}, \ldots, \vec{x_T}\}$ , the GMM likelihood can be written as

$$p(X \mid \lambda) = \prod_{t=1}^{T} p(\vec{x_t} \mid \lambda). \tag{2.4}$$

Thereafter, ML parameters can be estimated by using a specialized version of the expectation-maximization (EM) algorithm. The basic idea of the EM algorithm is, beginning with an initial model $\lambda$, to estimate a new model $\overline{\lambda}$, such that $p(X \mid \overline{\lambda}) \geq p(X \mid \lambda)$. The new model then becomes the initial model for the next iteration as the process is repeated until some convergence criterion is reached.

On each EM iteration, the following estimates are evaluated. These estimates guarantee a monotonic increase in the model's likelihood value:

Mixture Weights:

$$\vec{p_i} = \frac{1}{T} \sum_{t=1}^{T} p(i \mid \vec{x_t}, \lambda) \tag{2.5}$$

Means:

$$\vec{\overline{\mu_i}} = \frac{\sum_{t=1}^{T} p(i \mid \vec{x_t}, \lambda) \vec{x_t}}{\sum_{t=1}^{T} p(i \mid \vec{x_t}, \lambda)} \tag{2.6}$$

Variances:

$$\overline{\sigma_i}^2 = \frac{\sum_{t=1}^{T} p(i \mid \vec{x_t}, \lambda) \vec{x_t}^2}{\sum_{t=1}^{T} p(i \mid \vec{x_t}, \lambda)} - \overline{\mu_i}^2 \tag{2.7}$$

The *a posteriori* probability for acoustic class $i$ is given by

$$p(i \mid \vec{x_t}, \lambda) = \frac{p_i b_i(\vec{x_t})}{\sum_{k=1}^{M} p_k b_k(\vec{x_t})} \tag{2.8}$$

## 2.4 Algorithmic Issues

Two factors in training a Gaussian mixture speaker model are the order $M$ of the mixture and how to initialize the model parameters prior to the EM algorithm. There is no reliable theoretical method to guide one in either of these selections. Therefore, they are best experimentally determined for a given task. How to experimentally determine these two factors is discussed below.

**Initializing:** As we know, the EM algorithm can only be guaranteed to find a local maximum likelihood model. But the likelihood equation for a GMM has several local maxima and different initial models can lead to different local maximum. In [9], several initializing methods have been compared, and the result shows that the different initial models may converge to different local maxima of the likelihood function, but the difference between the final models is insignificant.

**Model order:** Determining the number of components $M$ in a mixture required to model a speaker adequately is an important but difficult problem. Choosing too few mixture components can produce a speaker model which does not accurately model the distinguishing characteristics of a speaker's distribution. On the other hand, choosing too many components may reduce performance when there are a large number of model parameters relative to the available training data and result in excessive computational complexity both in training and classification.

In [9], experiments that test the relation between model order and speaker verification system performance are proposed. The result shows that models must contain at least a minimum number of components to maintain good performance. This minimum number seems to be 16, and the increase in performance begins to level out above 32 component Gaussians.

## 2.5 General Speaker Verification Based on GMM

### 2.5.1 General Approach

The general approach proposed by Douglas Reynolds [10] for the speaker verification system is to apply a likelihood ratio test to an input utterance to determine if the claimed speaker should be accepted or rejected. Given an utterance $X = \{x_1, \ldots, x_T\}$, a claimed speaker identity with corresponding model $\lambda_C$ and an anti-model $\lambda_{\overline{C}}$, the likelihood ratio is defined by

$$\frac{\Pr(\text{X is from the claimed speaker})}{\Pr(\text{X is not from the claimed speaker})} \tag{2.9}$$

$$= \frac{Pr(\lambda_C|X)}{Pr(\lambda_{\bar{C}}|X)} \tag{2.10}$$

$$= \frac{Pr(X|\lambda_C)/Pr(X)}{Pr(X|\lambda_{\bar{C}})/Pr(X)} \tag{2.11}$$

Discarding the constant prior probabilities for claimant and imposter speakers, the likelihood ratio in the log domain becomes

$$\Lambda(X) = \log p(X|\lambda_C) - \log p(X|\lambda_{\bar{C}}). \tag{2.12}$$

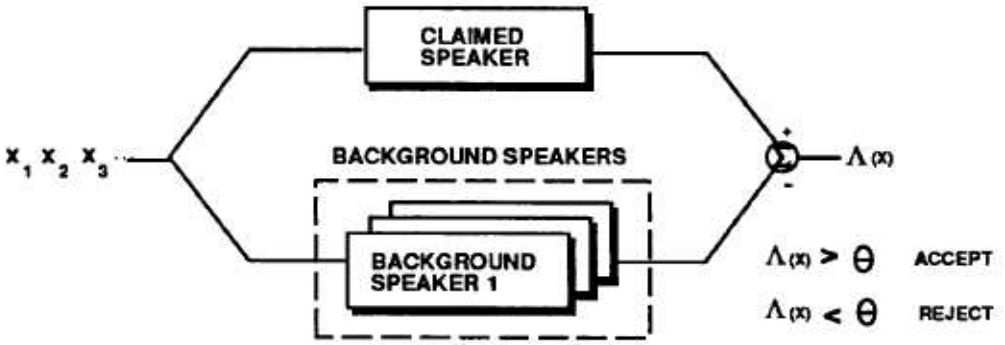


Figure 2.3: Speaker verification system using GMM[10]

The term $p(X|\lambda_C)$ is the likelihood of the utterance given it is from the claimed speaker and $p(X|\lambda_{\bar{C}})$ is the likelihood of the utterance given it is not from the claimed speaker. The likelihood ratio is compared to a threshold $\Theta$ and the test speaker is accepted if $\Lambda(X) > \Theta$ and rejected if $\Lambda(X) \leq \Theta$. The likelihood ratio essentially measures how much better the claimant's model scores for the test utterance compared to some non-claimant model. The decision threshold is then set to adjust the trade-off between rejection true claimant utterances (false rejection errors) and accepting non-claimant utterances (false acceptance errors).

The ways to compute the terms of the likelihood ratio are described as follows. The likelihood of the utterance given the claimed speaker's model is directly computed as

$$\log p(X|\lambda_C) = \frac{1}{T}\sum_{t=1}^{T} \log p(x_t|\lambda_C). \tag{2.13}$$

The likelihood of the utterance given it is not from the claimed speaker is estimated based on a collection of background speaker models. With a set of $B$ background speaker models, $\lambda_{\overline{C}} = \{\lambda_1, \ldots, \lambda_B\}$, the background speakers' log-likelihood is computed as

$$\log p(X|\lambda_{\bar{C}}) = \log\{\frac{1}{B}\sum_{b=1}^{B} p(X|\lambda_b)\}, \tag{2.14}$$

where $p(X|\lambda_{\bar{b}})$ is computed as in Equation (2.13). In [10], $B$ is set to ten. Except for the $\frac{1}{T}$ scale, this is the joint probability density of the utterance coming from one of the background speakers assuming equal-likely speakers. The use of background speakers to form various likelihood ratio tests has been used in several different speaker verification systems [4].

### 2.5.2 Background Speaker Selection

Two issues that arise with the use of background speakers are:

**1.The selection of the speakers.**

**2.The number of speakers to use.**

Intuitively, the background speakers should be selected to represent the population of the expected imposters, which is in general application specific. In some scenarios, we may assume that imposters will only come from similar sounding or at least speakers with the same gender. In a telephone based application , the imposters may sound very dissimilar to the users they attack

(casual imposters). Previous systems choose the background speakers by selecting those whose models are the closest or the most competitive for each registered speaker. This may be appropriate for the dedicated imposter scenario, but in [4], the experiments show that this makes the system exposed to imposters which have very dissimilar voice characteristics. This occurs because the dissimilar voice is not modeled well by either the numerator or denominator of the likelihood ratio. Therefore, when the imposter population contains dissimilar speakers from the users, the background selection should also include dissimilar as well as close speakers.

Ideally, the number of background speakers should be as large as possible to better model the imposter population, but practical considerations of computation time and storage prefer a small set of background speakers. The limited size was motivated by real-time computation considerations.

Douglas Reynolds proposed a systematic approach to the selection of the background speakers [10]. In this approach, GMMs of all speakers in the database are created using training data and pair-wise distances between the speaker models are computed. For speakers $i$ and $j$ with models $(\lambda_i, \lambda_j)$ and training utterances $(X_i, X_j)$, the distance is defined as

$$d(\lambda_i, \lambda_j) = \log \frac{p(X_i|\lambda_i)}{p(X_i|\lambda_j)} + \log \frac{p(X_j|\lambda_j)}{p(X_j|\lambda_i)}. \tag{2.15}$$

The ratio $p(X_i|\lambda_i)/p(X_i|\lambda_j)$ measures how well speaker $j$'s scores with speaker $i$'s speech relative to how well speaker $i$'s model scores with his/her own speech. The similar the models, the smaller the ratio becomes. The distance

measure is then a symmetric combination of ratios comparing models $\lambda_i$ and $\lambda_j$. Then, the followed procedure is applied to select two sets of background speakers, one set for the $B/2$ closest (similar) speakers, and the other for the $B/2$ farthest (dissimilar) speakers.

Let $\mathcal{F}_c(i)$ denote the $N$ closest speakers in the set of background speakers for speaker $i$ and $\mathcal{B}(i)$ denote the final $B$ background speaker set. Usually, $N$ is set to a number greater than $B/2$, and in [10], $N = 20$. $\mathcal{B}(X)$ is selected from $\mathcal{F}_c(i)$ by finding those $B/2$ which are maximally spread from each other. More specifically, the procedure is as follows:

**1.** Start by moving the closest speaker from $\mathcal{F}_n(i)$ to $\mathcal{B}(i)$.
Let $N = N - 1$, $B' = 1$ where $B'$ is the current number of speakers in $\mathcal{B}(i)$.

**2.** Move speaker $c$ from $\mathcal{F}_c(i)$ to $\mathcal{B}(i)$, where $c$ is found by

$$\begin{aligned} c &= \arg\max_{c \in \mathcal{F}_c(i)} \left\{ \frac{1}{B'} \sum_{b \in \mathcal{B}(i)} \frac{d(\lambda_b, \lambda_c)}{d(\lambda_i, \lambda_c)} \right\} \\ N &= N - 1, B' = B' + 1. \end{aligned} \tag{2.16}$$

**3.** Repeat Step 2 until $B' = B/2$.

The far speaker selection is accomplished by using the pair-wise distance measure as before, but now selecting the maximally spread 5 speakers from the $N$ farthest speakers, denoted as $\mathcal{F}_f(i)$ here. The $B/2$ background speakers are selected via the following procedure:

**1.** Start by moving the farthest speaker from $\mathcal{F}_f(i)$ to $\mathcal{B}(i)$.

Let $N = N - 1$, $B' = 1$.

**2.** Move speaker $f$ from $\mathcal{F}_f(i)$ to $\mathcal{B}(i)$, where $f$ is found by

$$\begin{aligned} f &= \arg \max_{f \in \mathcal{F}_f(i)} \left\{ \frac{1}{B'} \sum_{b \in \mathcal{B}(i)} d(\lambda_b, \lambda_c) * d(\lambda_i, \lambda_c) \right\} \\ N &= N - 1, B' = B' + 1. \end{aligned} \tag{2.17}$$

**3.** Repeat Step 2 until $B' = B/2$.

Finally, $\mathcal{B}(i)$ contains 10 ($B/2 + B/2 = |\mathcal{B}(i)| = B = 10$) background speakers, and the value of Equation (2.14) can be computed by the Gaussian mixture models of these background speakers.

### 2.5.3 Threshold

To make the verification system in Figure 2.3 complete, we have to obtain the threshold value. For training data $X = \{X_1, \ldots, X_n\}$, the threshold value is obtained by the following steps:

**1.** Calculate $\Lambda(X_i)$ by Equation (2.12), where $1 \leq i \leq n$.

**2.** Place $\Lambda(X_i)$ in one sorted list and set the point on the list at which the false acceptance rate (the percent of imposter tests above the point) equals the false rejection rate (the percent of true tests below the point) to be the threshold value.

Reynolds et al. [10] show that the “local threshold” has a better performance than “global threshold”. A global threshold measures the overall (speaker independent) system performance using the largest number of tests available, including both claimed speaker and imposter, and a local threshold treats each speaker's true and imposter utterances scores separately. But the “local threshold” has lower statistical significance due to the use of smaller number of tests available per speaker.

More recently, Reynolds et al. [8] proposed a new approach to selecting background speakers. In this new approach, a universal background model (UBM) is established from a large speech data set, ranging from 16 to 2048 speakers. Then for different claimed speaker, the EM algorithm is applied to adjust the weights of background speakers' models in UBM and the results can then be used in speaker verification. However, the authors admitted that ”there is no objective measure to determine the right number of speakers or amount of speech to use in training a UBM.” In other words, the performance of a speaker verification system still depends heavily on the quality of background speakers, but there is no systematic approach to the selection of background speakers.

# Chapter 3

# My New Approach: OSCILLO

As we see in the previous chapter, general speaker verification system with GMM needs a large background speaker database for training. A problem of their method is how to obtain the complete background speaker database so that every arbitrarily claimed speaker can get their best corresponding background speaker model. The ideal database may be huge, or it may take a lot of time to find background speakers. This thesis aims at a solution where we can construct the verification system using only the claimed speaker's speech data. We will describe the basic idea in Section 3.1 and details in subsequent sections.

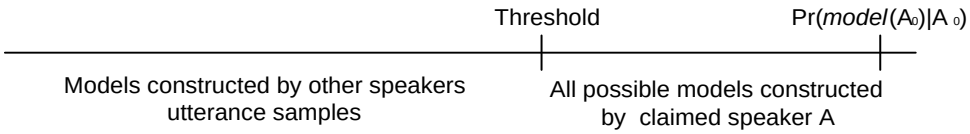


Figure 3.1: Preliminary

## 3.1 Preliminary

Given an utterance sample from claimed speaker $A$, that GMM is reliable, assuming the output score of GMM models created using $A$'s utterance should be greater than the score from a model of another speaker.

Figure 3.1 illustrates the idea of my new approach to speaker verification. The notation are defined as follows:

1. A claimed speaker $A$.
2. A specific utterance sample $A_0$ from $A$.
3. Model $model(A_i)$ constructed by sample $A_i$ .
4. Each point on the line is the posterior probability $\Pr(j \mid model(A_0))$ of $model(A_0)$ for the sample $j$.

The main assumption of the line in Figure 3.1 is that for the specific sample $A_0$:

$$\forall i, \Pr(A_0 \mid model(A_i)) > \Pr(A_0 \mid model(B)) \tag{3.1}$$

where $model(B)$ is the model constructed by another person's utterance sample and $model(A_i)$ is the model constructed by the claimed speaker's utterance.

If we can find the lowest value of $\Pr(A_0 \mid model(A_i))$, then we can take this value as a threshold to see if the test utterance belongs to the claimed speaker. Then, the problem becomes how to find the lowest value. In the next section, we will introduce a statistical technique, *tolerance interval analysis* [6], which helps us to find the lower bound of $\Pr(A_0 \mid model(A_i))$.

## 3.2 Tolerance Interval Analysis[6]

In this section, we will define tolerance regions in Subsection 3.2.1, and explain a simple method to construct them from a random sample in Subsection 3.2.2.

### 3.2.1 Definition

For any fixed region $R$ of a given population, we define the *coverage* of $R$ as the proportion of the population which lies in $R$, that is, the proportion of the population covered by $R$. In random variable terminology, the coverage of $R$ is

$$C(R) = P(X \in R), \tag{3.2}$$

where $X$ is drawn at random from the population.

Suppose that, for some purposes, we would ideally like to find a region with coverage 0.5, that is, a region including 50 percent of the population. Lacking special knowledge about the population distribution, we cannot accomplish this exactly. We might be willing, instead, to define a region so that there is a probability 0.95 that it will have coverage at least 0.5.

A *tolerance region* is a random region having a specified probability, say $1-\alpha$, that its coverage is at least a specified value, say $c$. Various names are given to $1-\alpha$ and $c$ in the literature. We shall call $1-\alpha$ the *confidence level* and $c$ the *tolerance proportion*, the latter because in some situations it is the minimum proportion of the population which it is considered tolerable to cover. We shall also speak of a "$c$ tolerance region with confidence $1-\alpha$." Regions which have this property under essentially no restrictions on the population are sometimes called "nonparametric tolerance regions," to distinguish them from "parametric tolerance regions," which have the required property as long as the population belongs to some specified parametric family, but not in general otherwise. Only nonparametric tolerance regions used here.

### 3.2.2 Construction of Tolerance Intervals: Wilk's Method

Assume that $X_1, \ldots, X_n$ are independent observations on the same distribution, with cumulative distributed function $F$. Let $X_{(1)}, \ldots, X_{(n)}$ denote the order statistics of these observations. Assume that $F$ is continuous, so that,

with probability one, there are no ties and a unique ordering $X_{(1)} < X_{(2)} < \ldots < X_{(n)}$ exists. Let $C_k$ be the coverage of the interval between $X_{(k-1)}$ and $X_{(k)}$ . Then by Equation (3.2) we have for $k = 2, \ldots, n$,

$$C_k = F(X_{(k)}) - F(X_{(k-1)}). \tag{3.3}$$

(Since $F$ was assumed continuous, the coverage is the same whether the endpoints are included in the interval or not. If a specific statement were required, we would assume that right (upper) endpoints are included, and left (lower) endpoints are not.) We further define

$$C_1 = F(X_{(1)}) \tag{3.4}$$

$$C_{n+1} = 1 - F(X_{(n)}) \tag{3.5}$$

as the coverage of the interval below $X_{(1)}$ and the interval above $X_{(n)}$ respectively. The definition in Equation (3.3) applies also to these two intervals once we define $X_{(0)} = -\infty$ and $X_{(n+1)} = \infty$.

We now have $n+1$ coverages, $C_1, C_2, \ldots, C_{n+1}$ , corresponding to the $n+1$ intervals into which the n sample points divide the real line. These $n + 1$ coverages are random variables, and their joint distribution has a number of interesting prosperities. A property which provides an easy method of construction of tolerance regions is the following. Let $i_1, \ldots,$ is be any s different integers between 1 and $n + 1$ inclusive: then the sum $C$ of the corresponding coverages,

$$C = C_1 + C_2 + \ldots + C_{i_s} \tag{3.6}$$

has the same distribution as the $s$th smallest observation in a sample of $n$ from the uniform distribution on the interval (0,1) , namely

$$P(C \geq c) = n \begin{pmatrix} n-1 \\ s-1 \end{pmatrix} \int_c^1 u^{s-1}(1-u)^{n-s} du \tag{3.7}$$

$$= \sum_{k=0}^{s-1} \begin{pmatrix} n \\ k \end{pmatrix} c^k (1-c)^{n-k} \tag{3.8}$$

The proof of the equivalence of Equation (3.7) and Equation (3.8) is given in Appendix B. Notice that the distribution depends on $s$ and $n$ only, not on which $s$ integers are chosen, nor on the distribution from which the sample was drawn, as long as $F$ is continuous. Notice also that $C$ has a beta distribution by Equation (3.7) and that Equation (3.8) is a left-tail binomial probability.

When we have $n$ samples, we will have $n-1$ coverages. Thus, $s = n-1$ and Equation (3.8) becomes:

$$\sum_{k=0}^{n-2} \begin{pmatrix} n \\ k \end{pmatrix} c^k (1-c)^{n-k}. \tag{3.9}$$

In Figure 3.2, we can see that as training sample size increases, the coverage increases when the confidence is fixed.

## 3.3 Training Procedure

Consider the size of the speech databases to be used in our experiment, we set 20 as the number of independent samples to obtain a tolerance region with coverage at least 0.8 and confidence level is 0.93. Each Gaussian mixture

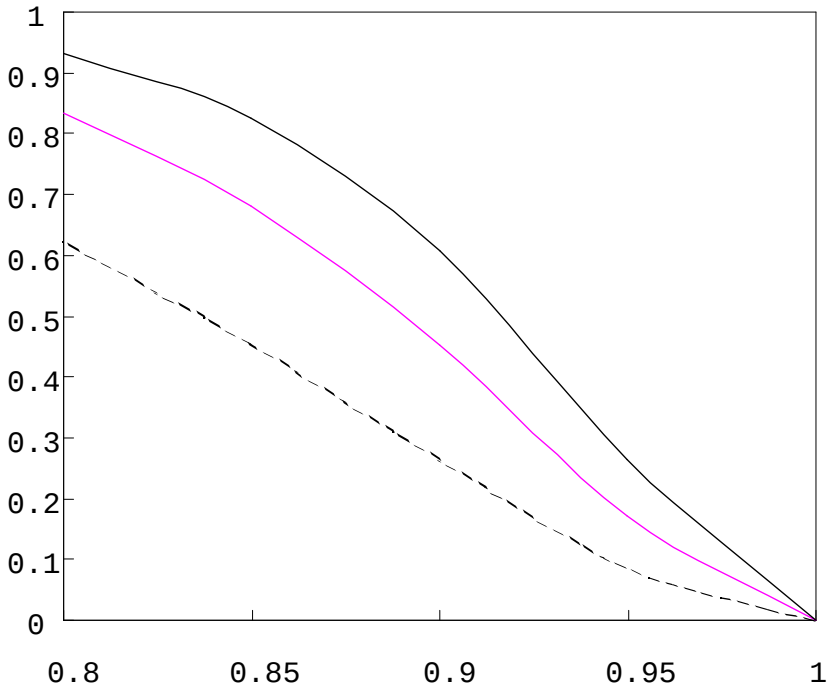


Figure 3.2: Tolerance interval

model is trained by 1000 feature vectors and 1000 feature vectors are extracted from about 10 seconds speech. Therefore, for the whole training system, we need approximately 3.5 minutes of speech from the claimed speaker to construct 20 GMM models. we denote the 20 samples as $\{A_0, \ldots, A_{19}\}$, and let $A_0$ to be the specific utterance sample. The training procedure is as follows:

1. Construct 20 Gaussian mixture models $\{model(A_0), \ldots, model(A_{19})\}$

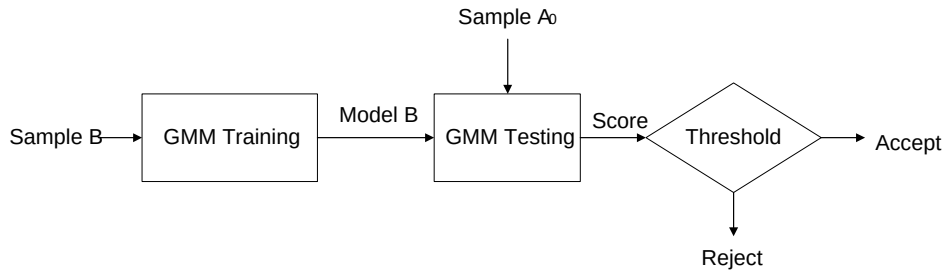


Figure 3.3: Verification procedure

from $\{A_0, \ldots, A_{19}\}$.

2. Calculate $\Pr(A_0 \mid model(A_i))$, $0 < i < 20$.

3. Let the minimum value of $\Pr(A_0 \mid model(A_i))$ to be our final threshold .

## 3.4 Verification Procedure

For the input test utterance $B$, we first construct a Gaussian mixture model $model(B)$ using this sample. Then calculate the posterior probability of $model(B)$ for the specific utterance sample $A_0$. Finally, compare the posterior probability to the threshold we just obtained. We take the test utterance as the utterance from the claimed speaker if the posterior probability is greater than the threshold and not from the the claimed speaker otherwise. Figure 3.3 shows the diagram of the verification procedure.

## 3.5 An Alternative Approach

An alternative approach is to obtain the threshold by gathering 20 samples from the claimed speaker, and calculate the posterior probability of each sample related to a specific claimed speaker model. Finally, as before, set the smallest value to be the threshold. For verification, simply estimate the posterior probability of an input utterance based on the same model and check if the result is greater than the threshold.

Unfortunately, the experiments show that it performs poorly. We will explain why this is the case with an example given in Table 3.5. In this table, we denote $A_0$ as the sample from the claimed speaker, and $B_0$ is the sample from the imposter. Each row in the table shows the output score given the same utterance, and the last column is the identity of utterance in each case. Since $A$ has a higher score than $B$ using $model(A_0)$, and vice versa, these two rows represent two successful speaker identification cases of GMM.

To apply the alternative approach in speaker verification, assume that we use $model(A_0)$ and the threshold is set to 0.6. Then in the first case, $X$ is an utterance sample of $A$, so $X$ can be verified correctly. But in the second case, since $\Pr(X \mid model(A_0)) = 0.65$ is greater than the threshold, the system will mistakenly verify that $B$ is the claimed speaker $A$, although it is a successful case in speaker identification.

Table 3.1: Why alternative approach works poorly

| | $\Pr(X \mid model(A_0))$ | $\Pr(X \mid model(B_0))$ | identity |
|---|---|---|---|
| $X$ | 0.7 | 0.6 | $A$ |
| $X$ | 0.65 | 0.8 | $B$ |

One may try to increase the threshold, but in our experiments, this only leads to higher false rejection rate. We concluded that a GMM model can accurately output posterior probabilities so that for different speakers these probabilities will show the relative likelihood that the input utterance is from a certain speaker. There might not exist an absolute value that allows us to draw a decision boundary with a threshold to verify whether the speaker is the claimed speaker accurately.

# Chapter 4

# Experimental Evaluation

## 4.1 Data Sets

We applied three databases for the experiments in this thesis. They are TCC300, NIST, and TIMIT. These three databases are widely used as a benchmark of speaker recognition research. The characteristics of these databases are shown in Table 4.1.

TCC-300 is a collection of microphone speech databases produced collaboratively by three universities in Taiwan: National Taiwan University, National Cheng Kung University, National Chiao Tung University. The speech data of each university are provided by 100 speakers (50 males, 50 females), and we use the data provided by National Chiao Tung University only. The sampling rate is 16kHz. A 30ms Hamming window was applied to the speech every 10ms.

Table 4.1: Databases for our experiments

| Database | number of speakers | number of speakers we used | Channel |
|---|---|---|---|
| TCC-300 | 300 | 100 | microphone |
| TIMIT | 630 | 168 | microphone |
| NIST | 1060 | 50 | telephone |

The TIMIT database allows examinations of speaker recognition performance under almost ideal conditions. Each speaker in this database contains 10 utterances and each utterance is about 3 seconds.

The full name of NIST database is 2000 NIST (National Institute of Standards and Technology) Speaker Recognition Evaluation Corpus [7]. The 2000 NIST Speaker Recognition Evaluation is part of an ongoing series of yearly evaluations conducted by NIST. The speech of each speaker is stored in one corresponding file which contains about 10 utterances.

## 4.2 Evaluation

We conducted some data preprocessing step on the databases including feature extraction and then combine them together in one text file. Therefore, every speaker in the database has one corresponding text file. The feature vectors are obtained by HTK tools [11]. Each feature vector consists of 13 features. The procedure to extract features from speech waveform is given in Appendix A. A silence removing procedure is applied to filter out vectors where log energy analysis result (the 13th feature) is below a threshold value.

Table 4.2: Threshold for silence

| | TCC-300 | TIMIT | NIST |
|---|---|---|---|
| Threshold | 25 | 12 | 15 |

This threshold is different for each database as shown in Table 4.2.

During the training procedure, we segmented all the vectors into 20 samples. Each sample contains 1000 feature vectors. But in TIMIT and NIST, the speech data per speaker is not enough for 20 segments. Each speaker in TIMIT has only about 2500 vectors, so we can get only 2 independent samples, and the rest 18 samples are obtained by select random start points on the vectors. Similarly, each speaker in NIST has about 9900 vectors. We have 9 independent samples and the remainder are obtained by the same way as for TIMIT.

The Gaussian mixture model is constructed using a C++ class library [3]. This library contains more than 20 classes including commonly used feature extraction algorithms and modelling techniques for speech recognition and speaker verification.

We divide the programming code into three parts:

1. Training: Getting the threshold value.
2. Testing part1: Test the data from the claimed speaker and get the value of false rejection rate.
3. Testing part2: Test the data not from the claimed speaker and get the

value of false acceptance rate.

In our experiments, the selection of the specific utterance sample may affect the performance of the system. The sample with noise will decrease the accuracy of the GMM based speaker identification system. If the accuracy decreases in the speaker identification case, our speaker verification system will not work well either. The speech in a microphone based database is quite clean and the system will not be affected no matter which sample is chosen. Unfortunately, it is not so ideal when we use the telephone based speech database. The selection of the specific utterance sample becomes important. We have to select the most suitable sample among the 20 samples by comparing the deviations. The selection procedure is as follows. We select each one of the 20 samples as the specific utterance sample iteratively and calculate the posterior probabilities of the models constructed by the remainder samples. We have 19 posterior probabilities each turn and the deviation of these values will be calculated. The deviation is obtained by the equation blow,

$$\begin{aligned} s &= \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - n\mu^2}{n}} \\ \mu &= \frac{1}{n}\sum_{i=1}^{n} x_i \end{aligned} \tag{4.1}$$

The smaller the deviation, the better the sample is. The one which has the smallest deviation will be chosen.

| Number of samples for training | FR | FA |
|---|---|---|
| 10 | 14.6% | 0% |
| 15 | 7.8% | 0% |
| 20 | 5.6% | 0.1% |

Table 4.3: Comparing different number of training samples.

## 4.3 Comparison

First, we compare the performance with different number of training samples. The result is shown in Table 4.3. To test false rejection rate (FR), 500 testing samples are used (5 samples each speaker). To test false acceptance rate (FA), 5 samples of each speaker except the claimed speaker are collected (total $100 * 99 * 5 = 49500$ samples). The result shows that as the number of samples increases, the false rejection rate decrease and the false acceptance rate does not increase significantly.

In the following subsections, we compare OSCILLO and the general approach using different databases.

### 4.3.1 TCC-300

In this experiment, the general approach is trained by 600 samples (6 samples from each speaker). The FR is tested by 300 test samples, and FA is tested by 29700 (100 claimed speakers*99 imposters*3 samples) samples. The value of OSCILLO approach is the same as Table 4.3.

| | FR | FA |
|---|---|---|
| General approach | 16.6% | 1.2% |
| OSCILLO | 5.6% | 0.1% |

Table 4.4: Experimental result use TCC-300 database.

| | FR | FA |
|---|---|---|
| General approach | EER=0.24% | |
| OSCILLO | 5% | 0.3% |

Table 4.5: Experimental result use TIMIT database.

### 4.3.2 TIMIT

In OSCILLO approach case, the FR is tested by 1680 (10 samples for each speaker)test samples, and FA is tested by 28056 (168 claimed speakers*167 imposters*1 samples) samples. The value of general approach is cited from [10].

### 4.3.3 NIST

The general approach is trained by 300 samples (6 samples from each speaker). The FR is tested by 100(50 claimed speakers*2 samples) test samples, and FA is tested by 49500 (50 claimed speakers*49 imposters*2 samples) samples. In OSCILLO approach case, the FR is tested by 500 (10 samples for each speaker) test samples, and FA is tested by 22050 (50 claimed speakers*49 imposters*9 samples) samples.

We find the result is not as good as shown in Renoylds' paper [10]. The

| | FN | FP |
|---|---|---|
| General approach (150 samples for training) | 24% | 6.5% |
| OSCILLO | 7% | 22% |
| OSCILLO(Select model) | 4.8% | 0.9% |

Table 4.6: Experimental result use NIST database.

| | TIME |
|---|---|
| General approach | 53.7minutes |
| OSCILLO | 59.65seconds |

Table 4.7: Comparing the time taken for training.

reason might be that the databases we use cannot offer good selections of the background speakers or the number of true and false data we offer to the general approach for training is unbalanced or insufficient. We also show the time taken for training and testing in Table 4.3.3 and Table 4.3.3 using TCC-300.

At last, we are going to compare the general approach and OSCILLO in the application of preventing private copy. When a customer buy a specific product, we wish that he can enroll his identity to our server before starting to use the product. Then we wish that only the owner of the product can use it. The system progress will be as follows and shown in Figure 4.1:

| | TIME |
|---|---|
| General approach | 20seconds |
| OSCILLO | 3.665seconds |

Table 4.8: Comparing the time taken for testing.

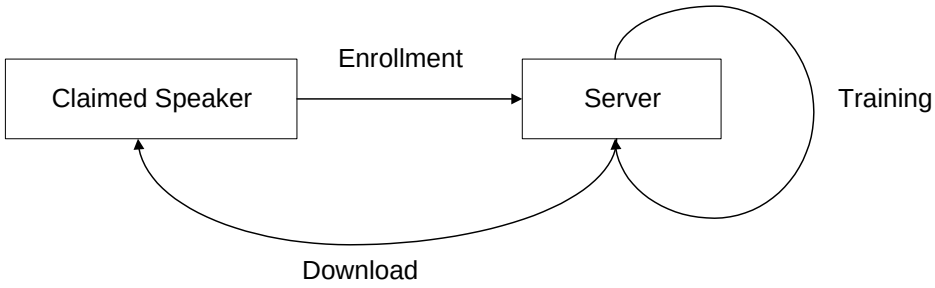


Figure 4.1: Progress of the preventing private copy system

**1.** The customer upload his speech file into our server.

**2.** The server progress the training procedure and generate some information. The information will be downloaded by the buyer and combined with the original product to accomplish the speaker verification system.

The information which has to be downloaded using the general approach includes the whole background models and the threshold. But using OSCILLO, the customer only need to download the threshold value . Apparently our approach, OSCILLO, is more efficient in this application.

# Chapter 5

# Applications

## 5.1 Potential Applications

Speaker verification has been applied in many domains. The most important domain is security applications. When speech of a visitor is available and security is an issue, speaker verification can be deployed to verify the visitor's identity. We will introduce some applications below:

i. Voice security system can be used for the quick access to doors, car protection against thefts, the protection of electronic systems like TV, video and the protection of computers.

ii. One of the biggest markets for speaker verification is secure access to services via the telephone, including home shopping, home banking and telecom services.

iii. Detection and tracking of a speaker in a (telephone) conversation between two or more speakers is a relatively new area of research, but has a large number of possible applications, including speaker indexing of large audio archives and real-time subtitling of TV broadcasts.

iv. In a forensic context, speaker verification can be deployed in the processing of telephone tapes, either to identify a talking suspect, or track the time intervals where a suspect is talking in a lengthy telephone conversation.

v. Another market is secure access to information which can be obtained through the internet.

vi. Monitoring is another important application. Speaker verification is applied to monitor the whereabouts of persons who are not allowed to travel freely, for example rehabilitating prisoners.

vii. preventing private copy that we have described in the last chapter.

Over the above are the applications of speaker verification system. Another important contribution of OSCILLO is that it can be used the solve the open-set problem in speaker identification system. Thus, the domains that speaker recognition system can be applied become more extensive.

## 5.2 Remaining Issues

The biggest problem we suffered is that the data of one speaker in presenting databases is usually insufficient. This problem also causes the inconvenient of claimed speakers that they have to speak more sentences to enroll their identity. A long utterance data is required for our testing is also the shortcoming that we need to overcome. In the testing procedure, we need the test utterance of about 10 seconds. But the general approach can easily get good performance using only a shorter test utterance of about 5 seconds or even 3 seconds.

All speaker recognition systems have two areas where future research can improve [8]. One is that the current systems rely on low-level acoustic information. But, speaker and channel information are mixed together and it is difficult to separate them from each other. The performance of the systems degrades when the microphone or acoustic environment change between training data and recognition data. The state-of-the-art channel equalization method can not solve the gap between matched and mismatched conditions.

The other area is incorporating higher levels of information, such as word usages, into the decision making process. The challenges are to find reliable extractions and efficient use of the higher-level of information from the speech signal.

# Chapter 6

# Conclusion

In this thesis, The open set classification based on tolerance interval for speaker verification system has been proposed. In fact, the speaker verification problem itself is an instance of open-set classification. The only difference is that the speaker verification system has just one class in the known set and in most open set classification applications, the known set usually contains many classes.

Setting a threshold value which determines whether the test utterance should be accepted or rejected is the most popular solution for the speaker verification system. OSCILLO provides a preliminary step toward the solution. It can be implemented easily and also has good performance. We hope that it can help the speech technology to be put in use widely earlier.

# Appendix A

# Feature Extraction

The input speech waveform has to be transformed into a sequence of parameter vectors. This process is required to as "*feature extraction*" and is critical to the performance of a speaker recognition. The process of feature extraction after years of studies in digital signal processing has been standardized [11] and The over all process is illustrated in Figure A.1 which shows the sampled waveform being converted into a sequence of parameter blocks. The input speech has to be sampled by the windowing process. Normally, the window size will be larger than the frame rate so that successive windows overlap as in Figure A.1.

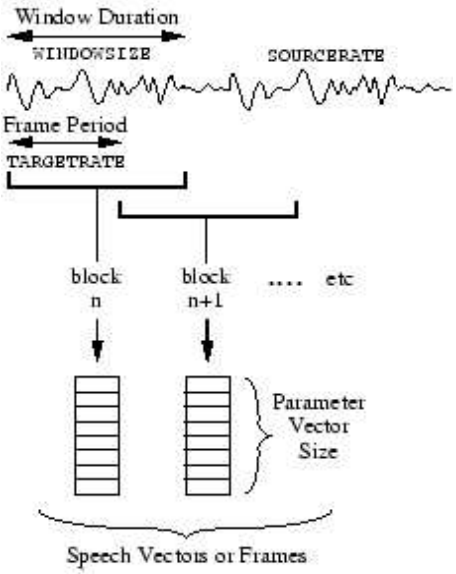


Figure A.1: Speech encoding process[11]

## A.1 Filterbank Analysis

The human ear resolves frequencies non-linearly across the audio spectrum and empirical evidence suggests that designing a front-end to operate in a similar non-linear manner improves recognition performance. A popular alternative to linear prediction based analysis is therefore filterbank analysis since this provides a much more straightforward route to obtaining the desired non-linear frequency resolution.

A simple Fourier transform based filterbank is designed to give approximately equal resolution on a mel-scale. Figure A.2 illustrates the general form of this filterbank. As can be seen, the filters used are triangular and

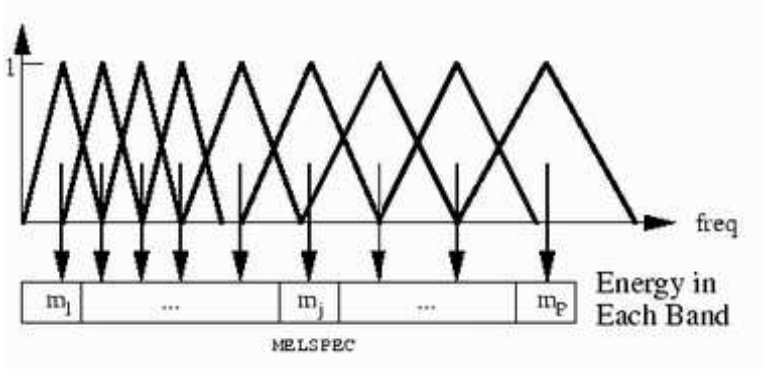


Figure A.2: Mel-scale filter bank[11]

they are equally spaced along the mel-scale which is defined by

$$Mel(f) = 2596 \log_{10}\{1 + \frac{f}{700}\}. \tag{A.1}$$

To implement this filterbank, the window of speech data is transformed using a Fourier transform and the magnitude is taken. The magnitude coefficients are then binned by correlating them with each triangular filter. Here binning means that each FFT magnitude coefficient is multiplied by the corresponding filter gain and the results accumulated. Thus, each bin holds a weighted sum representing the spectral magnitude in that filterbank channel.

# Appendix B

# Proof of Equation (3.7) and Equation(3.8)

$$
\begin{aligned}
P(C \geq c) &= n \left( \begin{array}{c} n-1 \\ s-1 \end{array} \right) \int_c^1 u^{s-1}(1-u)^{n-s} du && \text{(B.1)} \\
&= n \left( \begin{array}{c} n-1 \\ s-1 \end{array} \right) \int_c^1 u^{s-1} \frac{d(-(1-u)^{n-s+1})}{n-s+1} && \text{(B.2)} \\
&= n \left( \begin{array}{c} n-1 \\ s-1 \end{array} \right) \left\{ u^{s-1} \frac{-(1-u)^{n-s+1}}{n-s+1} \Big|_c^1 - \int_c^1 \frac{-(1-u)^{n-s+1}}{n-s+1} du^{s-1} \right\} && \text{(B.3)} \\
&= n \left( \begin{array}{c} n-1 \\ s-1 \end{array} \right) \left\{ c^{s-1} \frac{(1-c)^{n-s-1}}{n-s+1} + \frac{s-1}{n-s+1} \int_c^1 (1-u)^{n-s+1} u^{s-2} du^{s-1} \right\} \\
&= \frac{n(n-1)!}{(s-1)!(n-s+1)!(n-s+1)} c^{s-1}(1-c)^{n-s+1} && \text{(B.4)} \\
&\quad + \frac{n(n-1)!(s-1)}{(s-1)!(n-s)!(n-s+1)} \int_c^1 (1-u)^{n-s+1} u^{s-2} du^{s-1} \\
&= \frac{n!}{(s-1)!(n-s+1)!} c^{s-1}(1-c)^{n-s+1} && \text{(B.5)} \\
&\quad + \frac{n(n-1)!}{(s-2)!(n-s+1)!} \int_c^1 (1-u)^{n-s+1} u^{s-2} du^{s-1}
\end{aligned}
$$

$$= \begin{pmatrix} n \\ s-1 \end{pmatrix} c^{s-1}(1-c)^{n-s+1} \tag{B.6}$$

$$+n \begin{pmatrix} n-1 \\ s-2 \end{pmatrix} \int_c^1 u^{s-2}(1-u)^{n-s+1} du$$

$$= \sum_{k=0}^{s-1} \begin{pmatrix} n \\ k \end{pmatrix} c^k (1-c)^{n-k} \tag{B.7}$$

The second item in Equation (B.6) is continually decomposed by the previous steps, and finally we obtain Equation (B.7)

# Bibliography


[1] A. P. A. Broeders. The role of automatic speaker recognition techniques in forensic investigations. In *Proceedings of the International Congress of Phonetic Sciences*, pages 154–161, 1995.

[2] Sadaoki Furui. An overview of speaker recognition technology. In *Automatic Speech And Speaker Recognition*, pages 31–56. Chin-Hui Lee, Frank K.Soong and Kuldip K.Paliwal, 1996.

[3] Jialong HE. C++ class library (sv_lib), 1996.

[4] L. Bahler Higgins and J. Porter. Speaker verification using randomized phrase prompting. *Digital Signal Processing*, 1, 1991.

[5] L. G. Kersta. Voiceprint identification. *Nature*, 196:1253–1257, 1962.

[6] John W. Pratt and Jean D. Gibbsons. *Concepts of Nonparametric Theory, Springer Series In Statistics, Springer-Verlag*, 1981.

[7] Mark Przybocki and Alvin Martin. 2000 nist speaker recognition evaluation. *Speeh Communications*, 31, 2000.

[8] Douglas A. Reynolds Thomas F. Quatieri and Robert B.Dunn. Speaker verification using adapted gaussian mixture models. *Digital Signal Processing, Academic Press*, 2000.

[9] Douglas A. Reynolds. Robust text-independent speaker identification using gaussian mixture speaker models. *IEEE Transactions on speech and audio processing*, 3(1), Jan. 1995.

[10] Douglas A. Reynolds. Speaker identification and verification using gaussian mixture speaker models. *Speech Conmunication*, 17, 1995.

[11] Dan Kershaw Steve Young, Gunnar Evermann and Gareth Moore et al. *The HTK Book*, 2001.