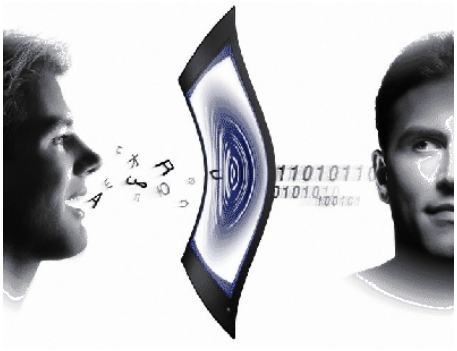

Aalborg University

Master Thesis project

Speaker recognition using universal background model on YOHO database

Author:
Alexandre Majetniak

Supervisor:
Zheng-Hua Tan

May 31, 2011

AALBORG UNIVERSITET

The Faculties of Engineering, Science and Medicine
Department of Electronic Systems
Frederik Bajers Vej 7
Phone: +45 96 35 86 00
http://es.aau.dk

Title:
Speaker recognition using Universal Background Model on YOHO speech database
Theme:
Digital signal processing
Project period:
February 1st - May 31st, 2011

Project group: 10gr926

Group members:
Alexandre MAJETNIAK

Supervisor:
Zheng-hua Tan

Number of copies: 3
Number of pages: 51
Appended documents:
(appendix, DVD)
Total number of pages: 54
Finished: june 2011

Abstract:

State of the art of Speaker recognition is fairly advanced nowadays. There are various well-known technologies used to process voice prints, including hidden Markov models, Gaussian mixture models, Vector Quantization
The goal of this project is first, to extract key features from a speech signal using MATLAB. Using MFCC as a feature extraction technique, the key features are represented by a matrix of cepstral coefficients . Then, using a statistical model and features extracted from speech signals, we build an identity for each person enrolling in the system. This paper presents a project using first, a Gaussian mixture models (GMM) as a statistical model for text independent speaker recognition, and secondly a universal background model, also called World model. GMM have proven to be effective for modeling speaker identity since it clearly represents general speaker-dependent spectral shapes. UBM improves GMM statistical computation for decision logic in speaker verification
Expectation and Maximization algorithm, an effective technique for finding the maximum likelihood solution for a model, is used to train speaker-specific and world model. This paper briefly presents advanced methods used to improve speaker recognition accuracy such as SVM and NAP. The experimental evaluation is conducted on the YOHO database composed of 138 speakers, each recorded on a high quality microphone. The system uses the large amount of input speeches from the speakers to train a universal background model (UBM) for all speakers and a model for each speaker. Many test speeches are provided to verify the identity of each speaker.

Preface

This report documents group 926's work on the 10th semester of the *Multimedia, interaction and signal processing* specialisation at the Institute of Electronic Systems, Aalborg University. The work was done during the period from February 1st to May 31st.

The report is divided into 5 parts: *Introduction*, *Feature extraction*, *Modeling*, *Testing and implementation* and *Test data and Evaluation*. The first part motivates the project and describes in an overview each step in speaker recognition and presents its different variants. Feature extraction details the first step of speaker recognition which consists in extracting features from speech data. The 3^{rd} part describes the second step of the process which consists in modeling. It presents mainly two different techniques used along this project: GMM and UBM. The 4^{th} part presents the testing phase, which is followed by the programming code description. Finally, the last part evaluates the system's performance and draws a conclusion.

A bibliography listing all the relevant literature sources can be found at the end of the report. The references are made using the syntax [number].

I would like to thank my supervisor at Aalborg University Zheng-Hua Tan for allowing me to work on this project, which was very instructive to me.

Alexandre Majetniak

Table of Contents

Part I

Introduction

Contents

This part of the report presents the motivation and need for speaker recognition system. It explains the most relevant speaker recognition systems already existing as well as the test part.

Chapter 1

Motivation

Speaker recognition systems have been studied for many years. They are nowadays widely used in several application fields. Speaker recognition can be defined as the process of recognizing the person speaking, based on people's speech recordings (speech waves), which provide information about each speaker. This method allows a speaker to use his voice as an identity verification for several purposes such as voice operators, telephone transactions, and shopping, information or database access, remote access computers, voice mail, security check for confidential information areas and remote access computers.

The goal of speaker recognition is mainly to facilitate the everyday life and replace most repetitive task, more particularly in the field of telephone shopping/bankind or information services. It is a strong security component for confidential areas access. For instance, a person's unique voice cannot be obtained in any way involving computer hacking skills, such as a password. The only possible harm would involve stealing a person's sample. Considering that security areas mainly use *text-dependent speaker verification* systems, an intrusion requires recording in a noisefree environment samples of an individual's voice, spelling a particular sentence. Therefore it is very unlikely to happen. Some speaker imitation systems exist and allow to record a person's voice to apply any speech in order to make them say anything, but these systems are still being developed. The most advanced one nowadays, are only used by powerful structures such as MI6, CIA, FBI etc.

In order to have an effective speaker identification system, a quality recording environment is required, with a set of training and testing data as large as possible. A more exhaustive speech database statistically increases the chance of a match during the test.

There are several other technical parameters to take into account, which alter matching speaker's effectiveness. These main matters will be discussed further on. The system used for this project has been developed using well-known state-of-the art function from speech processing researches.

First, we will brievly describe the existing speaker recognition process, then we will discuss mainly about each step of the process. On the latter, we will describe the YOHO database and its use for this project. Finally, the system's performance using YOHO database will be presented and discussed, which will provide an overview of the database's benefit along this project. Finally, we will provide a conclusion illustrating the main matter of speaker recognition.

1.1 Process description

Speaker recognition system are of two different kinds:

- text-dependent speaker recognition: The speaker is evaluated taking into account the pronounced text.
- text-independent speaker recognition: The speaker is evaluated disregarding of the pro-

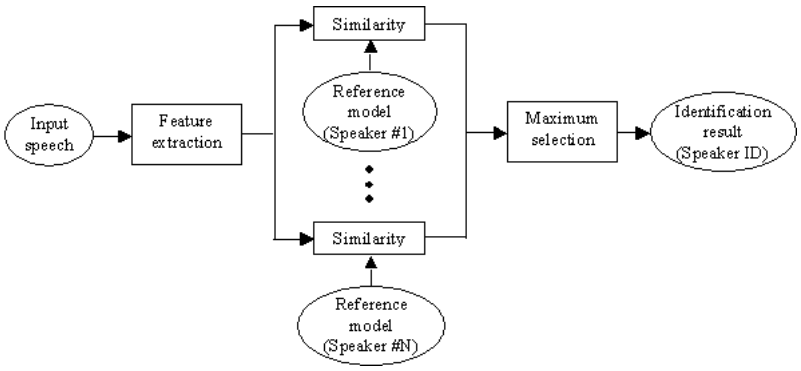

Figure 1.1: *speaker identification process*

[1]

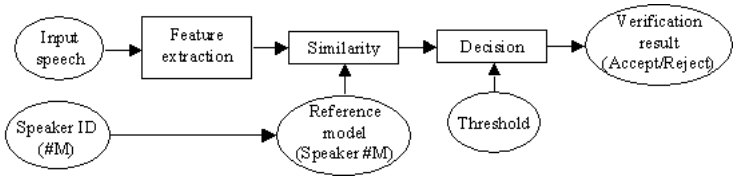

Figure 1.2: *speaker verification process*

[1]

nounced text.

The recognition process is separated into two different categories:

- **speaker verification**: The speaker claims his identity and the given speech is processed and compared to the training model corresponding to this speaker and the system determines if there is a match.
- **speaker identification**: The speaker provides a test speech which is processed and compared with each model of the training database. It results a log-likelihood computation for each speaker using the expectation-maximization algorithm. The higher score corresponds to the unknown speaker

Further on, speaker recognition systems have two main modules: *feature extraction* and *feature matching*. Feature extraction consists in extracting data(feature vectors) from the speech signal which will be processed later to identify each speaker. Feature matching involves recognizing the unknown speaker comparing extracted features from his or her voice with a collection of enrolled speakers

Above, a representation of the identification and verification modules, cf. Figure 1.1 and figure 1.2

Part II

Feature extraction

Contents

This part of the report presents the first necessary step in each text dependent/independent speaker recognition systems. When the speech data is first processed through a reader, the output data is too large to be processed and is suspected to be notoriously redundant, which implies: much data for few relevant information. It is composed of the sampling frequency F_s and the sampled data y . The latter needs to be transformed into a reduced representation set of features called *feature vectors*. This process is called *feature extraction*. When applying the appropriate method, the feature set, or reduced representation of the full size input, will hopefully be composed of relevant information in order to perform universal background modeling and speaker verification later.

Chapter 2

Mel-frequency cepstral coefficients

This section deals with the first techniques applied in order to use the input utterance. First, we need to extract speech features. Using digital signal processing tools (DSP), the operation consists in converting the speech waveform into a set of features called an *acoustic vector*. This is more commonly called, the signal-processing *front end*. The output of an MFCC is a feature vector. On Figure 2.1 an example of a speech signal.

When observing the signal waveform on a short period of time, we recognize similar patterns, the speech characteristics are quasi-stationary. On the other hand, when the chosen period exceeds 1/5 seconds, the patterns will change according to the various speech sounds produced by the speaker's voice. Therefore, it is more common to use *short time spectral analysis* to characterize a speech signal.

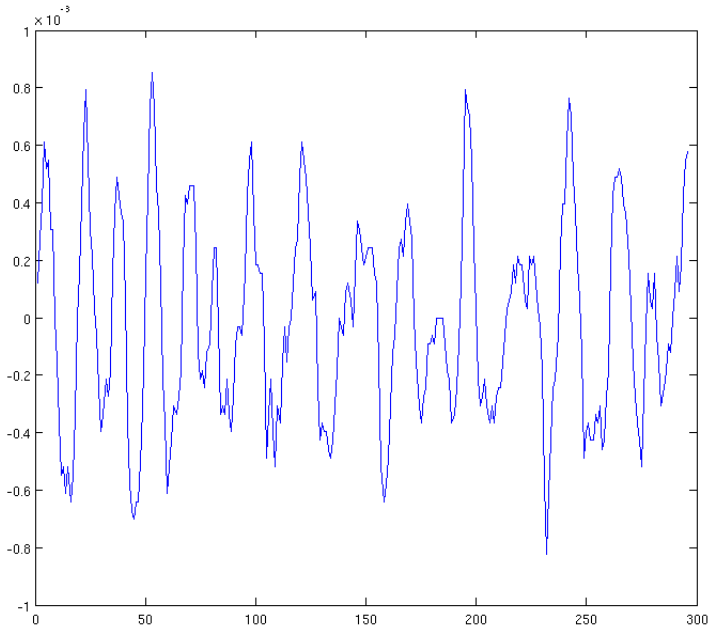

Figure 2.1: *Example of a speech signal*

Besides MFCC, there are several different techniques existing to parametrically represent a speech signal. The latter are for instance, Linear Prediction Coding (LPC), which is a previously used technique widely exploited in field of speaker recognition. However, this model-based representation can be strongly affected by noise. MFCC uses a computed filterbank applied on the frequency domain, which can considerably reduce noisy speech recognition. Nowadays, it also remains the most widely used technique, therefore it is used along this project.

The principle of the MFCC technique is to simulate the behavior of the human ear. It operates on the known range of the human ear's bandwith. A fixed number of frequency filters are applied on the signal and distributed on the full range. On low frequencies, the filters are applied linearly, whereas the high frequencies use logarithmic filter. The purpose of using filters is to capture the phonetically important features of a speech by getting rid of the irrelevant ones. The first linear frequency filter is applied below 1000Hz and the second applied logarithmically above 1000Hz. This representation is defined as the *mel-frequency* scale.

Below, a scheme representing the structure of an MFCC.

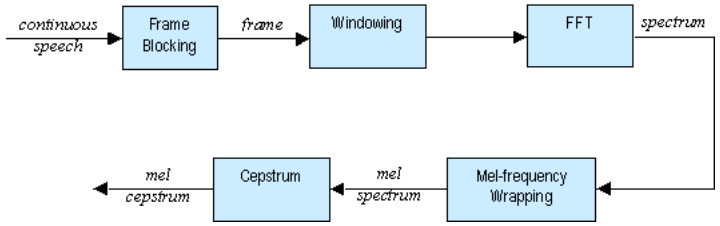

Figure 2.2: *Block diagram of the MFCC structure [1]*

When extracting features from an input speech, the goal is to reach a compromise between the acoustic vector's dimensionality and its discriminating power. As a matter of fact, the bigger its dimensionality, the more it requires training and test vectors. In the other hand, a small dimensionality is less discriminative therefore will not be an effective speaker identification/verification system.
Extracted features must therefore satisfy some conditions such as:

- Speech features must be easy to exploit
- Distinguish between speakers while maintaining a reasonnable threshold in order not to bee too discriminiative nor not enough.
- Features must be unsensitive to mimicry
- Environmental change between recording sessions must be minimal.
- Over time changing voice characteristics should not affect the set of features considerably.

2.1 MFCC process

The MFCC process is composed with several steps. First, frame blocking takes as an input the continuous speech (wavefile), and converts it into several frames of N samples. The processing operations generate fluctuations in between the frames. These irregularities are observed at the beginning of each frame, as well as at the end. Therefore, the next step is to window each individual frame in order to reduce this effect. Then, the Fast Fourier Transform starts the conversion from time to frequency domain for each frame of N samples, which outputs a result reffered as spectrum or also called periodigram. The next step is called *Mel-frequency warping*. The objective is to use a filter bank which filters the signal in the frequency domain. The number of filters is arbitrarly chosen and each are distributed uniformly on the *Mel-frequency scale*. A threshold value of 1000Hz

determines a change in the scaling type. Below threshold, the frequency spacing is linear, whereas it becomes logarithmic above. The frequency gap produced by a pitch variation above 1000Hz is much higher than an identical pitch variation below 1000Hz, hence the idea of a threshold. Finally, the process reaches the final step.

The log mel-spectrum is converted back to time, which outputs a set of *cepstrum* coefficients, also called an *acoustic vector*.

The MFCC process consists in taking an entire speech utterance as an input, producing a set of acoustic vectors, each of them having a dimensionality fixed by a number of cepstrum coefficients (usually 12).
The sampling rate F_s equals 12500Hz.

2.1.1 Frame Blocking

The frame-blocking process consists of separating the speech signal into frames of N samples. The second frame starts M frames after the first, with $M \leq N$. Consequently, it overlaps the first with a range of $N - M$ samples. The third frame overlaps the second with the same amount of samples, and so on until the process reaches the end of the speech signal, considering one frame or more must be produced. Usually N=256 and M=100, which corresponds to an overlaping of 156 frames for each frame.

2.1.2 Windowing

Following the frame blocking process, the speech signal encounters fluctuations at the edges of each frame. They refer as spectral distortion. The windowing process consists in reducing the signal discontinuities by lowering down its value to zero at the edges of each frame.

Analytically, assuming a arbitrary window represented as follows: $w(n)$,$n = 0...N - 1$, where N is the number of samples in each frame, the process of windowing will result in:
$y_i(n) = x_i(n)w(n)$, $n = 0...N - 1$ [1]

The hamming window will have the following form:

$$w(n) = 0.54 - 0.46cos(\frac{2\pi n}{N-1})$$

, $n = 0...N - 1$ [1]

2.1.3 Fast Fourier Transform (FFT)

The following step consists in converting each frame into the frequency domain. The corresponding operation is called Discrete Fourier Transform (DFT). Several algorithm have been developed to implement the DFT. Our interest turns to a fast and effective algorithm called Fast Fourier Transform (DFT). The transform is represented as a set of N samples x_n, below:

$$X_k = \sum_{i=0}^{N-1} X_n \exp \frac{-j2\pi kn}{N}$$

,$k = 0, 1, 2, ..., N - 1$ [1]
X_k's are complex number but we are only concerned with the real and not the imaginary component of the complex numbers, more precisely their absolute value,which in our case, corresponds to the frequency magnitude. The resulting sequence X_k can be explained the following way:

The values located in the following range: $n = 0...\frac{N}{2} - 1$ correspond to the actual frequencies: $f = 0...\frac{F_s}{2}$. Analytically, values of n such that $n = \frac{N}{2} + 1...N - 1$ correspond to the range of frequencies $f = \frac{f_s}{2}...0$, F_s defines the sample frequency.

The output of the following section is called spectrum or periodigram.

2.1.4 Mel-Frequency Wrapping

Psychosocial studies revealed that human perception does not follow a linear scale. The variation in frequency between two tones is bigger in high frequencies than in the lower frequencies. Mel-Frequency Wrapping consists in using a filter-bank which will measure a subjective pitch for each tone, on the mel-frequency scale. "The mel-frequency scale is a linear spacing below 1000Hz and a logarithmic spacing above 1000Hz."

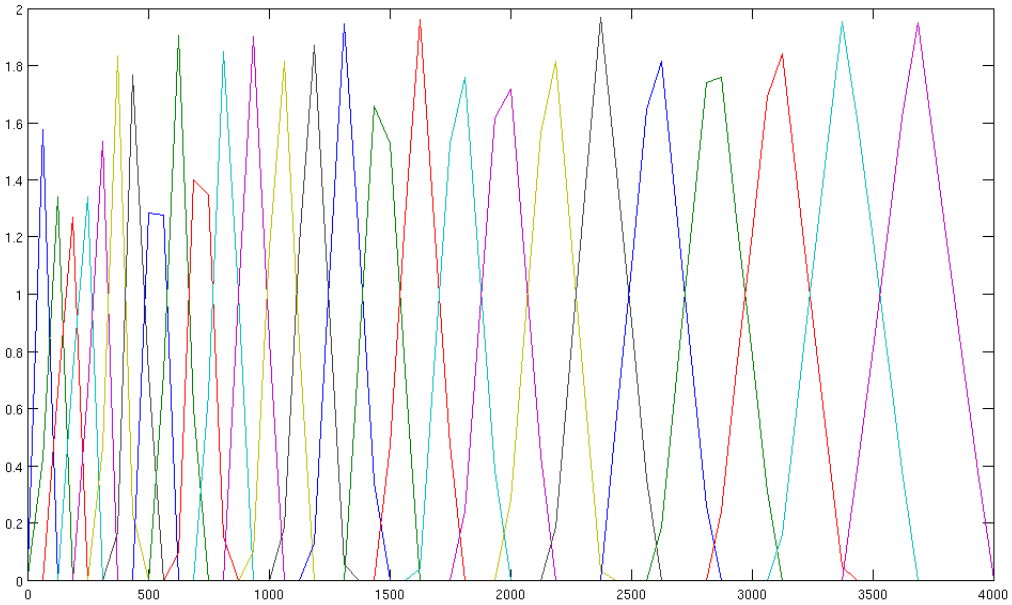

Figure 2.3: *Mel-spaced filterbank*

The filter bank attributes several bandpass filter on the scale. Each filter has a triangular shape, for which upper and lower cut off frequencies (bandwidth) are given by a constant mel frequency interval. This value also defines the spacing. The dimensionality of the acoustic vector, which corresponds to the number of cepstral coefficients is chosen as 12 on default configuration. The triangle shape windows are applied to the spectrum in the frequency domain.

Below, a representation of the idealized mel-space filterbank, without output sampling.

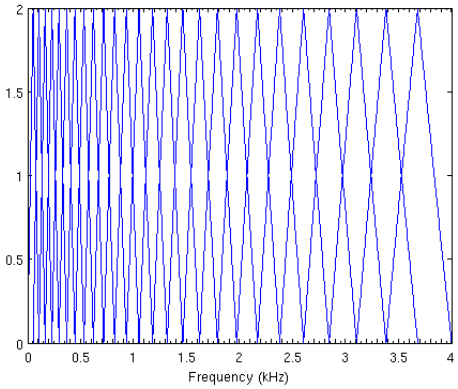

Figure 2.4: *Idealized Mel-spaced filterbank*

2.1.5 Cepstrum

The last step of the process consists in converting back the log-mel spectrum to the time domain. In order to perform this, we use the Discrete Cosine Transform (DCT), which will output the mel-frequency cepstrum coefficients (MFCC). The spectral properties can be observed efficiently when using the speech spectrum. First, it provides the mel-spectrum coefficients. Taking the logarithm of the mel-spectrum coefficients, we obtain real numbers, which can therefore be converted into the time domain. Let's assume a set of mel spectrum coefficients: $\tilde{s_0}, k = 0, 1, ..., K-1$, [1] the mel-frequency cepstrum are calculated as followed:

- For each mel-spectrum coefficients, take the power, then take the log of the resulting values.
- For each mel log powers, Apply the discrete cosine transform
- Extract the amplitude of each resulting cosine transform (spectrum), which is defined as the mel-frequency cepstral coefficients (MFCC).

The resulting equation of the MFCC is:

$$\tilde{C_n} = \sum_{k=1}^{K} (\log \tilde{S_k}) \cos(n(k - \frac{1}{2})\frac{\pi}{K})$$

[1]

Chapter 3

Other feature extraction methods

In the previous chapter, we described the main steps of the MFCC method as a feature extraction technique. It exists several other tools which allow feature extraction. Those are for instance *Linear Predictive coding* (LPC), or its variant *Warped linear predictive coding*.

3.1 Linear predictive coding

Linear predictive coding is an encoding method for speech processing. It is based on the linear predictive model. In such a model, the values are estimated as a linear function of the previous ones. It works as a sequence. The LPC method considers that a buzzer generates the speech signal. The buzzer located further away in the throat is responsible for the various types of sounds. A sound is subdivided into several components, accounting voiced sounds, which possess representative vocal characteristics, such as vowels. It also contains consonnants or whistling, whispering sounds, produced with a bigger amount of air in the voice. Those attributes compose an appropriate model for a good approximation of a speech production. The buzz or vibration is produced by the glottis. The glottis characteristics its volume and pitch (frequency). The vocal tract and the mouth composes the vocal tube. Consonnants, sibilants are produced by the movements of the tongue and lips, touching the teeths and the inside area of the mouth.
The role of LPC is to estimate particular components of a frequency spectrum of speech sounds. These are called "formants". The interaction between formants outputs distinct characteristics of vowels and consonnants. The resonnance of the tube generates the formants.
Resuming, a speech signal is composed with the following characteristics:

- the buzzer, produced by the glottis (denoted by its frequency and intensity)
- the tube, produced by the throat and the mouth (vocal tract), outputing the formants (components of the speech signal)
- sibiants and consonnants (lips and tongue)

The next step of LPC consists "inverse filtering" the speech signal by removing the formants. This result in substracting the tube specific sounds to the original speech signal. The remaining filtered speech is called the residue. The original signal is divided into three distinct parts: the residue signal, the formants and a set of numbers resulting from the buzz's frequency and intensity parameters.

After isolating the different attributes, LPC creates a source signal using the buzzer and the residue. This source signal is filtered using the formants, which outputs a speech signal. Alike the MFCC technique, LPC operates on a sequence of frames, with a general frame rate of 30 to 50 frames/sec. Using small speech extracts such as frames retains periodicity. It allows to avoid the signal's variation with time.

3.1.1 The prediction model

This section presents a technical overview of the prediction model. One most common representation is:

$$\hat{x}(n) = \sum_{i=1}^{p} a_i x(n-i)$$

[2] where $\hat{x}(n)$ is the predicted signal value, $x(n-i)$ the previouly observed value, and a_i the predictor coefficients. This estimate generates an error which is expressed as:
$e(n) = x(n) - \hat{x}(n)$ where $x(n)$ [2] is the true signal value.

these equations are valid for a system comprising only one dimension. In digital signal processing, the extracted features from a speech sample consist of several vectors of n dimensions.
For multi-dimensional signals, the error rate is expressed as:

$e(n) =\| x(n) - \hat{x}(n) \|$ [2]

3.1.2 Parameters estimation

The objective is to optimize the parameter a_i. The common choice in optimization is called the *autocorrelation* criteria. The method aims at minimizing the squared error expected value: $E[e^2(n)]$, which leads to the equation:

$$\sum_{i=1}^{p} a_i R(i-j) = -R(j)$$

, for $1 \leq j \leq p$ [2], where R is the autocorrelation of signal x_n, defined as

$$R(i) = E\{x(n)x(n-i)\}$$

[2] and E is the expected value.

3.2 Warped linear predictive coding

Waped linear predictive coding is a variation of the inner LPC algorithm. The main difference between them remains in the system's spectral representation. One solution consists with using "allpass filters" instead of 'unit delays" commonly used in LPC. An all-pass filter, alike with the concept of a low or high-pass filter, allows all frequencies to "pass". The only changes lies within the "phase response", which corresponds to the delay applied on the frequencies. The delay applied in an "all-pass" filter corresponds to a quarter of wavelength.
The main interest in using this technique, compared with standard linear predictive models, lies in the spectrum frequency resolution, which is rather closer to the frequency resolution of the human ear. Consequently, Warped LPC provides a higher accuracy in terms of speech feature extraction. [3]

Part III

Modeling

Contents

This part of the report presents the step following feature extraction. As described previously, the speaker recognition process is composed with two main phases: *enrollment* and *verification*. At the end of the enrollment phase, all speaker's voice utterances produce a series of features, which later form a voice print, template or model. On the verification phase, a speech (or several speech) samples are compared against all previously created voice print to determine the best match, hence recognizing the unknown speaker.

Chapter 4

Gaussian mixture model

The following section aims at describing the Gaussian mixture model and emphasize its use in the field of speaker recognition system. The previous section aimed at extracting features from an input audio speech using the Mel-frequency cepstral coefficient method(MFCC).
The GMM algorithm takes as an input a sequence of vectors provided by the MFCC and uses it to create one model per speaker, which is called the Gaussian mixture model.
In this section, we will describe the Gaussian mixture model and its parameterization.
First, the Gaussian mixture model is a "mixture density", characterized as a sum of M component densities. Each component density is a product of a "component Gaussian" with a "mixture weight". Each individual component Gaussians represent *acoustic classes*. [4] These classes reflect specific vocal tract configuration proper to a speaker and are therefore, useful for modeling speaker identity.
Second, a gaussian mixture density provides a good estimation independently speaking of the time differences between recording sessions. In other words, the GMM is not susceptible to natural vocal changes provoked by several factors such as aging or when a given speaker gets a cold.

4.1 Gaussian mixture model estimation

The Gaussian mixture density consists in a sum of M weighted component densities, given by the following equation

$$p(\vec{x}) = \sum_{i=1}^{M} p_i b_i(\vec{x}) \tag{4.1}$$

[4]

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

$$b_i(\vec{x}) = \frac{1}{(2\pi)^{\frac{D}{2}} \|\Sigma i\|^{\frac{1}{2}}} \exp -\frac{1}{2}(\vec{x} - \vec{\mu_i})' \Sigma^{-1} (\vec{x} - \vec{\mu_i})$$

[4]

- $\vec{\mu_i}$ is the mean vector extracted from feature matrices
- $\sum i$ is the covariance matrix which provides information about the difference between features.

The mixture weights are normalized and their sum must equal 1: $\sum_{i=1}^{M} p_i = 1$. [4] A component Gaussian is a function a mean vector with a covariance matrix. The product of a component

Gaussian with its respective mixture weight compose the component density. A sum of component densities defines the Gaussian mixture density. The mixture density parameters are defined as: $\lambda = \{p_i, \vec{\mu}_i, \Sigma i\} i = 1...M$. [4]

In the further step of identification, λ is used as the model of a speaker. Each speaker is attributed a GMM. Obtaining the appropriate λ for each speaker corresponds to the training phase.

The use of GMM can take several forms. The model may follow one of the 3 rules presented as followed:

- The model uses one covariance matrix per Gaussian component, also called: nodal covariance.
- The model uses one covariance matrix for all gaussian components in a speaker model: grand covariance
- The model uses a single covariance matrix shared by all speaker models: global covariance

[4]

In this specific case, the model has one covariance matrix per Gaussian component. Most implementations of gmm estimation functions use nodal covariance matrix, given that initial experimental results indicated better performance with this actual technique.

4.2 Uses of GMM and understanding the process

There are two important motivation in using Gaussian mixture densities for speaker identification systems.

The Component densities of a mixture models together a set of acoustic classes. The speaker's voice can be interpreted as an acoustic space which is characterized by a set of acoustic classes. They contain relevant phonetic characteristics of the speaker's vocals such as vowels, nasals and consonnants. In other terms, these acoustic classes provide several speaker-dependent vocal tract configurations, which makes them very useful for speaker identity.

The variables μ and Σ contain the following information about the acoustic classes:

- The mean $\vec{\mu_i}$ represents the spectral shape of the i^{th} acoustic class
- The covariance matrix Σ_i represents the variations of the average spectral shape.

Any training or testing speech is not labeled, therefore the acoustic classes are hidden. The purpose is to draw an observation from the hidden acoustic classes using the set of feature vectors. The resulting is called the observation density which is the Gaussian mixture. From one feature vector is produced a single mixture density. The sum of Gaussian mixture densities extracted from the set of feature vectors gives the GMM likelihood, which is the relevant attribute that allows us to further identify the unknown speaker.

The second motivation relies in the fact that the Gaussian mixture model is powerful in order to obtain accurate approximations with arbitrarily-shaped densities, which makes it more robust for speaker identification than other systems.

The design of the GMM emerges from two different models previously conceived.

- The classical unimodal Gaussian speaker model represents a speaker's distribution by a position, referred as the mean vector, and an elliptic shape being the covariance matrix.
- Vector quantization (VQ) defines a speaker's feature distribution by a set of characteristic templates.

The GMM is at the crossing of the two models. It combines both features from them by using a set of gaussian components, each of them depending on a specific mean and covariance matrix. This method provides a better modeling capability. On the bottom figure, we can observe a comparison of densities obtained using a unimodal Gaussian model, a VQ and finally, A GMM.

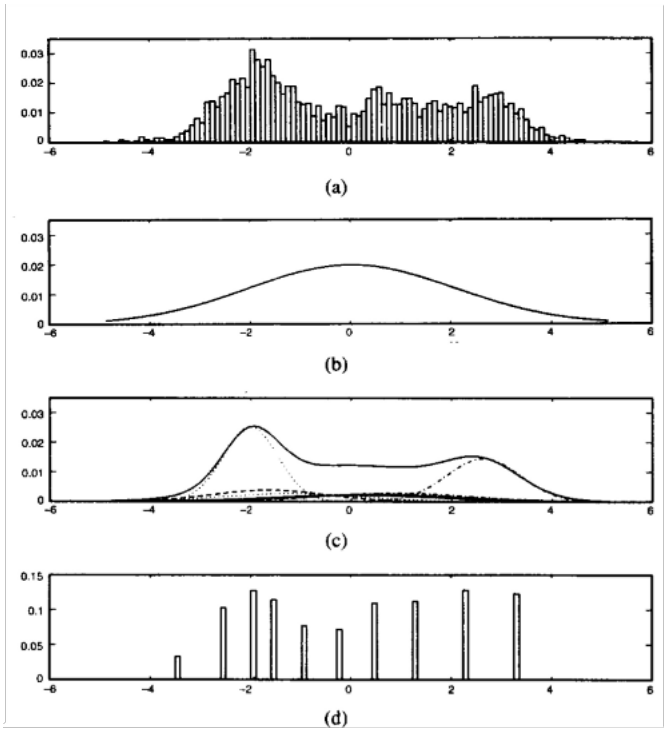

Figure 4.1: *Comparison of a distribution modeling: (a) Histogram of a single cepstral coefficient from a 25 second utterance by a male speaker; (b) max- imum likelihood unimodal Gaussian model; (c) GMM and its 10 underlying component densities; (d) histogram of the data assigned to the VQ centroid locations of a loelement codebook. [4]*

This analysis emphasizes the GMM nature as a combination of both unimodal Gaussian model ad Vector quantization. VQ generates a small distribution composed with 10 codebooks. The GMM provides a continuous and consequently bigger and much more accurate distribution. The distribution's shape underlines the density's "multi-modal" nature. Covariance matrices can be used in two ways: full or diagonal, but diagonal covariance matrices are shown to be more effective for speaker models. Full covariance matrices are therefore not necessary. Combining linearly diagonal covariance matrices can model the correlations between feature vector elements, since covariance matrices provide the variations between feature vectors.

Also, the use of a large diagonal covariance matrix is equivalent to using a small set of full covariance matrix.

4.3 Maximum likelihood parameter estimation

Holding a distribution of feature vectors, the goal of the training phase is to estimate the model λ that matches best this distribution. The technique used for this purpose is called *Maximum likelihood estimation.* In other terms, ML aims at finding the model parameters($\lambda = \{p_i, \mu_i, \Sigma_i\}, i = 1...M$ which maximize the likelihood of the GMM, given the training data (as shown in [4] and [5]).

Let there be the following sequence of vectors: $X = \{\vec{x_1}, ..., \vec{x_T}\}$. The GMM likelihood is written as:

$$p(X|\lambda) = \prod_{t=1}^{T} p(\vec{x_t}|\lambda)$$

[4]
The goal is to obtain Maximum-likelihood(ML) parameter estimates. The process is an iterative calculation called the expectation-maximization (EM) algorithm. The algorithm's name is rather explicit since the principle is:

- Beginning with an initial model λ, estimate an new model $\overline{\lambda}$ such that $p(X|\overline{\lambda}) \geq p(X|\lambda)$. The new model then becomes the initial model for the next step and so on. The model is recalculated iteratively, using the previous step to estimate the actual one. The process continues until a convergence threshold is reached, that is until the parameters of λ reach a stable value. The number of iteration often turns around 10, which is a generally accepted number of iteration for the algorithm the model to reach the threshold value.

On each EM iteration, the parameters are updated. Below are written the respective update formula for each parameter:

Mixture Weights

$$\overline{p_i} = \frac{1}{T} \sum_{t=1}^{T} p(i|\vec{x_t}, \lambda) \tag{4.2}$$

Means

$$\overline{\vec{\mu_i}} = \frac{\sum_{t=1}^{T} p(i|\vec{x_t}, \lambda)\vec{x_t}}{\sum_{t=1}^{T} p(i|\vec{x_t}, \lambda)} \tag{4.3}$$

Variances

$$\overline{\sigma_i^2} = \frac{\sum_{t=1}^{T} p(i|\vec{x_t}, \lambda)\vec{x_t}}{\sum_{t=1}^{T} p(i|\vec{x_t}^2, \lambda)} - \overline{\mu_i}^2 \tag{4.4}$$

[4] where σ_i^2, x_t, and μ_i refer to arbitrary elements of the vectors $\vec{\sigma_i}^2$, x_t and $\vec{\mu_i}$ $\{i = 1...M\}$ with M referring to the number of gaussians.

The last step of maximum-likelihood is to obtain the *a posteriori* probability for each feature vectors. From the equation (1), and using Bayes's a posteriori rule, we obtain *a posteriori* probability for an acoustic class i:

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

[4]

Chapter 5

Universal background modeling

Universal background model (UBM) is an improvement in the field of speaker recognition using Gaussian mixture models. It is used for speaker verification systems. It is typically characterized as a single Gaussian mixture model trained with a large set of speakers. As described in Reynolds's paper [6], the method is to first select a speaker-specific trained model, then determining a likelihood ratio of the match score of a test speech sample with the trained model and the universal background model. The latter recognizer is called GMM-UBM, [7] and uses *Maximum a posteriori* estimation. It consists of using UBM for training of the speaker-specific model. The first section describes the likelihood ratio, while the second describes the principle and uses of UBM. The third section provides analytical content of UBM and the last aims at discussing various alternative methods in using UBM, as well as speed-up recognition techniques.

5.1 Likelihood ratio

The likelihood ratio is defined as follows: Given an observation "O" and a hypothetical person P, the goal is to determine whether O is from P or not. Let us assume the two hypotesis:

- H_0 : O is from P
- H_1 : O is **not** from P

The likelihood ratio allows us to decide between the two hypothesis:

$$\frac{p(O|H_0)}{p(0|H_1)} \begin{cases} \geq \theta & H_0 \text{ is accepted} \\ \leq \theta & H_0 \text{ is rejected} \end{cases}$$

$p(O|H_0)$ is the probability of the hypothesis H_0 given the observation O. $p(O|H_1)$ is the probability of the hypothesis H_1 given the observation 0. In speaker verification, H_0 represents the hypothesis of a test speech utterance corresponding to a given training model. The background model is a non-speaker-specific model, and thus Hypothesis H_1 represents H_0's conjugate hypothesis: "the test speech does not correspond to its training model" Similarly, every test speeches compared to the background model yields the hypothesis: "Does not correspond to its model". Hypothesis H_0 and H_1 correspond respectively to a given model λ_p and its conjugate $\lambda_{\overline{p}}$. Using the latter hypothesis, the universal background modeling consists in calculating the likelihood from both hypothesis and computing the likelihood ratio described above.

The decision process depends on a threshold value corresponding to a given likelihood ratio. When the likelihood ratio goes over the given threshold, the hypothesis is accepted. Should the opposite occur, the hypothesis is rejected.

$$LR(X) = \frac{p(X|\lambda_p)}{p(X|\lambda_{\overline{p}})}$$

5.2 Interpretation of the UBM

A universal background model is a speaker-independent world model. It represents speaker independent distribution of the feature vectors used to form the model. It is trained with a huge amount of speech data (several hours) from a pool of speakers, using the EM algorithm. When a speaker enrolls into the system, the UBM is updated with speaker-independent features from the new speaker. Additionally, the adapted UBM is used as the target speaker model. This method prevents from having to build the speaker model (estimating the parameters) from scratch, usually with speech data instead. There are several ways to adapt the UBM. It is possible to adapt one or more of its parameters, as well as all parameters. Past experience has demonstrated that adapting "means only" is rather sufficient, Reynolds [6]. Adapting the means is done using the MAP (Maximum *a posteriori* method).
UBM is a GMM-based model; it acts as a large GMM, composed with a big amount of mixtures. When creating the model, one must take into account several parameters such as quality of the speech, composition of speakers. The background model must be built with speeches sharing common characteristics in type and quality. For instance, a system verification system using only telephone and male speakers must be trained using only telephone speech and male speakers. For a system where the gender composition is an unknown parameter, the model will be trained using male and female speeches. As shown in [6], it is important to have a uniform distribution or a good balance between male and female, otherwise, the model will bend towards the dominant population and alter the results. Similarly, other subpopulation are affected as well, such as microphone recording quality. For instance,using different types of microphones bends the dominance towards the most used type.
For male and female composition, one solution is to combine two UBMs, when one is trained with male and the second with female speakers. This technique solves the problems for unbalanced subpopulations.

5.3 Analytical process

As indicated previously, adapting only the means shows effective results, [7]. Given the enrollement feature vectors $X = \{x_1, ..., x_T\}$ and the UBM, $\lambda = \{P_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ the adapted mean results in:

$$\mu'_k = \alpha_k \tilde{x_k} + (1 - \alpha_k)\mu_k$$

[7]
where

$$\alpha_k = \frac{n_k}{n_k + r}$$

[7]

$$\tilde{x_k} = \frac{1}{n_k} \sum_{t=1}^{T} P(k|x_t)x_t$$

[7]

$$n_k = \sum t = 1^T P(k|x_t)$$

[7]
As mentioned in section 5.2, the MAP algorithm is used to derive a speaker-specific model from UBM. When performing speaker recognition, on technique consists in coupling both speaker-specific and background model for performance. The resulting recognizer is called *GMM-UBM*. The match score, described in section 5.1 depends on both the target λ_{target} and the background model λ_{UBM}.

The following average log likelihood formula gives a deeper level of abstraction from the one on section 5.1, from which it corresponds to.

$$LLR_{avg}(X, \lambda_{target}, \lambda_{UBM}) = \frac{1}{T}\sum_{t=1}^{T}\{log(p(x_t|\lambda_{target})) - log(p(x_t|\lambda_{UBM}))\}$$

[7]

where $X = \{x_1, ..., x_T\}$ corresponds to the set of observation or test feature vectors.

The higher the score, the more the test features are likely to belong to the speaker-model from which they are compared to. The use of background model gives a clearer match score range between the different speakers, and makes it more comparable that way, [7].
To improve performances, it is common to apply normalization on test segments and backgroung.

5.4 Alternative adaptation methods and speed-up recognition techniques

Other techniques aside of MAP exists to adapt speaker-specific GMM from UBM. One fairly used is called *Maximum likelihood linear regression* (MLLR) (Leggetter and Woodland, 1995) and shows effective results for short enrollment utterances, [7]. GMM, is heavy computationally due to frame-by-frame processing. GMM-UBM seeks for each test-utterance vector, the "top C" scoring Gaussians [8]. The speed can be improved by reducing the number of speaker models, or vectors.

Chapter 6

An overview on state of the art for speaker recognition

In the previous chapter, we discussed Universal background modeling and demonstrated its effectiveness for speaker verification. UBM is combined with GMM to extract the average log likelihood ratio. Techniques to speed up the system such as reducing number of speaker models or feature vectors, have been applied. However, state of the art in speaker recognition emerged new techniques to improve robustness with the use of classifiers like Support vector machine, combined with Nuisance Attribute projection (NAP) to compensate cross-channel degradation [9]. Newest technologies include Factor analysis (used in the ALIZE platform described in chapter 9, model compensation. This chapter first overviews other old technologies used to process and store voice prints. These include frequency estimation, hidden Markov models, Gaussian mixture models, pattern matching algorithms. Finally, it describes two recent state of the art techniques which are *Support vector machine* and *Nuisance attribute projection*.

The Gaussian mixture model is considered one of the most effective algorithm for speaker identification. However, there are various technologies used to process and store voice prints. These include frequency estimation, hidden Markov models, Gaussian mixture models, pattern matching algorithms. Some systems also use "anti-speaker" techniques, such as cohort models, and world models. This chapter aims at briefly discussing some of these techniques, their benefit and eventually providing a concise comparison with the GMM. [10]

6.1 frequency estimation

Frequency estimation is the process of estimating the complex frequency components of a signal in the presence of noise. In this sense, it provides more robustness to noise compared to GMM. The noise component of the speech signal is unknown, providing it can be of different type, intensity and distributed irregularly. Frequency estimation technique estimates the noise component such as solving for eigenvectors. [11]. Eigenvectors are called the non-zero vectors, in the sense that when multiplied by a given matrix, the result remains proportional to the original vector and change only in magnitude $\implies$ *Multiplicating an eigenvector with a matrix* $\iff$ *Multiplicating an eigenvector with a scalar* λ. The mathematical expression of this idea is as follows: A being a square matrix, a non-zero vector v is an eigenvector of A if there is a scalar λ such that:

$$Av = \lambda v$$

Eigenvectors variations are linear. Within a signal, noise components are only changing in magnitude, therefore, Eigenvectors reveal the presence of noise components. The method consists in subtracting the noise from the input to get an approximation of the signal of interest and finally, decomposing that signal in a sum of complex frequency components. In other words, the last step

allows "noise-free" voice of a given speaker to be reduced to a more manageable representation, which is the voice's peak of intensity on a few frequency components. This method happens to be effective when background noise is important.

Several well-known methods allow to extract the frequency components by identifying noise subspace. Those estimation techniques comprise Pisarenko's method, MUSIC, the eigenvector solution, and the minimum norm solution. [11]

Let us consider a signal, x(n), consisting of a sum of p complex exponentials in the presence of white noise, w(n). This may be represented as

$$x(n) = \sum_{i=1}^{p} A_i e^{jnw_i} + w(n)$$

Thus, the power spectrum of x(n) consists of p impulses in addition to the power due to noise.

6.2 hidden Markov models

Prior to defining the hidden Markov model, it is necessary explaining the basic Markov model [12], known as the *Markov chain*. The Markov chain defines a system model with a random variable changing through time. Consequently, the Markov property implies that the state of a variable only depends on the previous state.

A hidden Markov model consists in a Markov chain for which part of the state is observable, which consequently outputs observations giving little information in determining the system state. The interest in having access to a partial state is, to focus on the sequence of states, rather than on each state separately. Such a model is constantly making transitions from the current state to the next at rates, and with probabilities, determined by the model's parameters. When making a transition, the model emits an output with a known probability. The same output can be generated by a transition from multiple states, with different probabilities. In the particular case of speaker recognition, a hidden Markov model emits outputs representing phonemes with probabilities that depend on the prior sequence of visited states. A speaker uttering a sequence of phonemes (i.e., talking) corresponds to the model visiting a sequence of states and emitting outputs corresponding to the same phonemes. This method works well to authenticate the speaker by having him utter a sequence of words forming complete sentences.

Many Hidden-markov model based algorithm have been developed. Among them are the Viterbi algorithm, which computes the most-likely corresponding sequence of states. Another called the Baum-Welch algorithm estimates the starting probabilities, transition function and observation function of a hidden Markov model. The Hidden Markov is known as a good tool for speaker-dependent recognition on isolated words, continuous speech and phones. It provided decent results in each cases.

6.3 pattern matching algorithms

This last technique [13], is among the most complex used for speaker recognition and compares two voice streams: the one spoken by the authenticated speaker while training the system, and the one spoken by the unknown speaker who is attempting to gain access. The speaker utters the same words when training the system and, later, when trying to prove his identity. The computer aligns the training sound stream with the one just obtained (to account for small variations in rhythm and for delays in beginning to speak). Then, the computer discretizes each of the two streams as a sequence of frames and computes the probability that each pair of frames was spoken by the same speaker by running them through a multilayer perceptron–a particular type of neural network trained for this task. This method works well in low-noise conditions, and when the speaker is uttering exactly the same words used to train the system. This method stands for speech-dependent speaker recognition systems. It is perfect for secure access areas, as it is

considered a non-compliant system, in the way that the speaker's utterance is required a rigourous precision, as the aim is to restrain access to unauthorized persons.

6.4 Support vector machine

Support vector machine (SVM) is a powerful discriminator and is therefore mainly used for speaker verification, [7]. It offers great robustness and can be combined with GMM for performance. SVM is a classifier, which separates speaker-specific features from the background. It models the decision between two classes. One class represents the training feature vectors of the target speaker (the speaker-specific features). The second class represents the training feature vectors of different speaker, which is considered as the background. The first class is labeled "+1", the second is labeled (-1). After labeling the feature vectors accordingly, the role of SVM is to compute the equation of a hyperplane which orientation maximizes the margin separating the two classes. This way, the speaker-specific features and the background are clearly separated before they can be modeled using GMM.

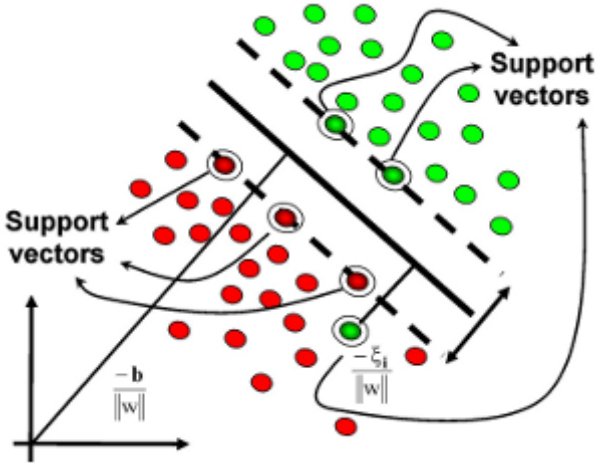

Figure 6.1: *A maximum margin hyperplane that separates positive and negative training features, [14]*

6.5 Nuisance attribute projection

Nuisance attribute projection (NAP) [9] is a technique used to reduce the "nuisance attributes" in classifiers, which is caused by the difference in audio quality recordings. For instance, a speaker using a microphone will output a different audio recording than someone using a phone. The difference in channel types causes nuisance attribute value. NAP aims at fixing the nuisance by either trying to find the nuisance attribute value or using "projection".

The principle of using projections is to use a projection matrix which removes the component of a feature vector in the direction of a specified subspace. The actual subspace must be the one containing informations about the channel. Isolating the channel subspace gives the possibility to compensate the nuisance.

Part IV

Testing and implementation

Contents

From this point, most major steps of the speaker recognition process have been treated. A set of feature vectors per speaker have been extracted using the MFCC method, which has been processed later by the expectation-maximization algorithm to create a gaussian mixture model per speaker. The original YOHO speech database has now turned into a set of mixture models. The last step comes with speaker identification, which we are describing in the first chapter. The second chapter is dedicated to the implementation and presents the enrollment and testing phase on MATLAB code.

Chapter 7

The identification process

The process consists in using a set of test speeches, applying the same method to extract a mixture per unknown test speaker and compare its model to each models of the training database. From each comparison between a test and a training model, is yielded a likelihood. The comparison which yields the highest score corresponds to the unknown speaker.
The following explanation describe technically the process:

Let us assume a group of speakers $S = 1, 2, ..., S$ represented by GMM's $\lambda_1, \lambda_2, ..., \lambda_S$. The objective is to find the speaker model which has the maximum a posteriori probability for a given observation sequence. In other terms, for a given set of n test feature vectors is extracted a set of n acoustic class, which provides us a set of n *a posteriori* probability, from which the maximum value provides us information for speaker identification.

The unknown speaker is therefore represented by:

$$\hat{S} = \arg\max_{1 \leq k \leq S} p(\lambda_k|X) = \arg\max_{1 \leq k \leq S} \frac{p(X|\lambda_k)Pr(\lambda_k)}{p(X)}$$

The equation can be simplified, taking into account two facts:

- All speakers are equally likely to be identified to the unknown speaker $\implies Pr(\lambda_k) = \frac{1}{S}$. Therefore, $Pr(\lambda_k)$ is a constant and can be neglected, as it does not affect effectiveness in finding the argument of the maximum a posteriory probability.
- All given observation sequence is likely to be identified with one of the models $\implies p(X)$ is the same for all speaker models. Therefore, $p(X)$ is a constant and can be neglected. The classification rule simplifies to

$$\hat{S} = \arg\max_{1 \leq k \leq S} p(X|\lambda_k)$$

Representing X as a set of vectors $X = x_1, x_2, ..., x_t$, $t = 1...T$, and using the logarithms, the unknown speaker model is identified to each model:

$$\hat{S} = \arg\max_{1 \leq k \leq S} \sum_{t=1}^{T} log(p(\vec{x_t}|\lambda_k))$$

Chapter 8

VOICEBOX matlab toolkit and programming code

This section deals with the necessary tools used for building a proper speaker recognition system. For this purpose, we have been using the VOICEBOX toolkit provided by Mike Brookes, Department of Electrical And Electronic Engineering from London, UK. State of the art in speech processing is already very wide and advanced, which provide us with several tools in front-end processing and speaker recognition which comprise "Audio File input/output", "Mel-frequency cepstral coefficients" "gaussian mixture model estimation", "log-likelihood computation".

The code was first developed using several training and testing features from the YOHO database. It was later adapted to use every training and testing data from the database.

This section aims at describing the entire process of speaker recognition generated by the matlab code. The comments on the code resume each step of the process. Later, we will describe each function, emphasizing the role they play in the recognition process.

Below, each main steps of the programming code are presented and described:

training.m:

...
$[s, fs] = wavread(filePath);$
...

- The **wavread** function takes as an input the path of a WAV file and returns the sample data "s" and the sample rate "Fs". The speech signal is expressed as one single supervector containing N samples data.

...
$m = melcepst(s, fs);$
...

- The **melcepst** function calculate the mel cepstrum of a signal. It takes as an input the speech signal, the sample rate in Hz as mandatory arguments. By default, it calculates the mel cepstrum with 12 coefficients and 256 sample frames. The output mel-cepstrum is a bi-dimensional cepstral coefficient matrix, or a set of feature vectors, which dimensionality is equal to the number of cepstral coefficients, hence, a data matrix (LxT), with L= number of cepstral coefficients (usually 12) T = number of feature (acoustic) vectors.

...
$[mu, sigma, c] = gmm_estimate(trainingFeatures\{i\}', 64, 10);$
...

The gmm estimate function outputs an estimation of the Gaussian mixture model λ. It takes as an input the column data matrix, which is the sequence of feature vectors output by the melcepst function, the number of gaussians (64, by default 12) and the number of iterations (by default 10). The initial means, diagonal covariance matrices and weights are initialized automatically.

8.1 The Expectation-Maximization(EM) algorithm

1. **FOR** 10 iterations:
   - estimation step: estimating the new model λ such that $p(X|\overline{\lambda}) \geq p(X|\lambda)$
   - Calculate Multigaussian Log likelihood (μ, σ, w)
   - Calculate mean log likelihood
   - maximization step: finding parameters of λ which maximizes the likelihood
   - Calculate new gaussian weights
   - calculate new variance matrix
   - Update

8.2 Test process using one test speaker

1. **IF** first input speech
2. **THEN** the test feature matrix is created
3. **ELSE** We concatenate the features of the next input utterance with the previous one
4. $score = -1000$ The likelihood is initialized to a very low number
5. **FOR** all training data
   - Calculate the test speaker multigaussian likelihood given the training model($\lambda = \mu, \sigma, c$).
   - $LLH = mean(IY)$ Calculate the mean of the likelihood vector and extract the model likelihood
   - **IF** $Likelihood \leq score$
     - $score = LLH$ The speaker with the highest likelihood is assigned to the unknown speaker
   - Display the likelihood
6. Display speaker!

8.3 MATLAB code structure using the full YOHO database

The MATLAB code has been written for two modeling technique. The first only uses GMM for speaker identification and aims at computing average log likelihoods (LLH_{avg})from each test data-set with each enrollment speaker models. At the end of the computation, a matrix of LLH_{avg} is produced and is used to calculate the false acceptance rate.
The second uses Universal Background Model (UBM) for speaker verification. The process is explained further on the corresponding subsection.

8.3.1 using GMM

This section describes precisely the final matlab code using GMM, optimized for computing the various results. Each step of the speaker identification process have been segmented to isolate the main time consuming computation parts from the analysis. This method is convenient in the sense that it allows to save significant amount of time for analysis.

training.m

The "training" function outputs a vector of Gaussian mixture models. The output vector is stored into the variable 'estimate'. Given the entire speech YOHO database, the estimate variable is composed with 138 Gaussian mixture models. Below, a résumé of the training process:

1. **FOR** each ENROLL speaker directory
    - **FOR** each session directory
        - **FOR** each speech utterance
            - read the wavefile and extract the signal and sampling frequency
            - extract the features using MFCC
            - Add the speech utterance features to the speaker training feature matrix (concatenation)
        - Add the speaker training feature matrix to the set of feature matrices
    - estimate the model parameters lambda using the gmm estimate function, with 24 Gaussians and 10 iterations
    - Add the model parameters to the set of models
2. return the set of mixture models

loadTests.m

The "loadTests" function's purpose is to extract the feature matrices from all speakers using all speech utterances in each sessions.

1. **FOR** each VERIFY speaker directory
    - **FOR** each session directory
        - **FOR** each speech utterance
            - read the wavefile and extract the signal and sampling frequency
            - extract the features using MFCC
            - Add the speech utterance features to the speaker training feature matrix (concatenation)
        - Add the speaker training feature matrix to the set of feature matrices
2. return the set of feature matrices

getAllLLH.m

The "getAllLLH" function takes as an input argument the set of mixture models (training), and the set of feature matrices (test). Then it computes the mean log-likelihood between each test feature matrix with each training model. There are exactly 138 models and the same amount of test feature matrices, which corresponds to 19044 mean log-likelihood computation (mLLH). The computation time allocated for each mLLH varies according to the parameters chosen. It will be discussed further in the chapter Performance evaluation 11.

1. **FOR** each test feature matrix
    - **FOR** each training model
        - Compute multigaussian log likelihood of the ith test feature matrix with jth training model
        - Compute mean log-likelihood (mLLH)
        - display mLLH
        - Add mLLH to the matrix of mLLH
2. Return the set of mean log-likelihood

testWithAllLLHinput.m

The "testWithAllLLHinput" computes the error rate based on a matrix of likelihood obtained from comparing each enrollment speaker to each test speaker. The "error rate" is computed based on the number of "false acceptance", see chap 11 for more detailed explanations about performance values.

1. initiate variable *numOferrors* to 0;
2. **FOR** each feature matrix
    - initiate variable *score* to $-9 \exp 99$
    - **FOR** each training model
        - **IF** $mLLH > score$
            - $score = LLH$
            - The unknown speaker is assigned to the present (jth) speaker
        - display mLLH
    - display the highest score and the number of the speaker concerned
    - **IF** the name of the identified speaker matches the name of the training feature
    - **THEN** display "speaker found"
    - **ELSE** display "false identification" and increment variable *numOferrors*
3. Compute the error rate, which equals to the quotient of the number of errors by the number of models, multiplicated by 100
4. display the error rate

8.3.2 using Universal Background Model

On the following section, there is no need to re-iterate the previous functions, which are rather similar to the section using GMM. The verification process is detailed below. It performs the following action for each speaker: *determining a likelihood ratio of the match score of a test speech sample with the trained model and the universal background model.* The process is re-iterated for a threshold value starting from 0.6 to 1.1, which are the lowest and highest threshold value, giving respectively the highest false rejection and highest false acceptance. For each threshold value, the program outputs the *false acceptance* and *false rejection* rate. The threshold value which gives the *equal error rate* is retained.

Verify-using-UBM-and-speaker-models.m

1. $threshold = 0.6$
2. **WHILE** $threshold \leq 1.1$
   - initialize number of *false acceptance* and *false rejection* to 0.
   - **FOR** each test feature file
     - **FOR** each training model
       - Compute likelihood ratio from i^{th}test using j^{th}training model and i^{th} UBM
       - Display likelihood ratio
       - **IF** $LLR \leq threshold$
       - **THEN** Accept the speaker
         - **IF** accepted speaker IS NOT the target speaker
         - **THEN** increment number of *false acceptance*
       - **ELSE**
         - Reject the speaker
         - **IF** rejected speaker IS the target speaker
         - **THEN** increment the number of *false rejection*
   - Display number of *false acceptance*
   - Display number of *false rejection*
   - Display *false acceptance rate*
   - Display *false rejection rate*
   - increment threshold (+0.01)
3. plot the *false rejection rate* as a function of *false acceptance* with the function $y = x$ to determine the *equal error rate*

Chapter 9

ALIZE library and LIA toolkit

Prior to using the matlab code as a toolkit for implementation of speaker recognition, a previous system has been worked on previously, called LIA SpkDet toolkit. Due to several problems in manipulating the code for the use of the YOHO database, the LIA platform was aborted along this project. The present chapter aims at describing the LIA-ALIZE platform, giving brief information about its content.
The first section aims at discussing the composition of the ALIZE library. The second section describes the LIA SpkDet toolkit and its use of the ALIZE library. The last section presents technical aspects in running the program.

9.1 The ALIZE library

The ALIZE platform has been developed at the laboratoire d'informatique d'avignon (LIA), by Frederic Wils, and directed by Jean Francois Bonastre since February 2003. It is composed with two distinct levels:

A first level which contains the different modules's levels of complexity (data acquisition, computation, storage, etc.). This level mainly prevents the user from managing memory allocation by himself. A second level includes utilities and algorithms manipulated by the user (list management, model initialisation, MAP algorithms, ...)

The ALIZE platform is composed with several data and computation servers, as shown in [15]. It is segmented in 4 individual components, presented in hierarchical order, as followed:

- A Data audio server which stores raw speech data coming from an input source (microphone) or a file audio source (wav, riff, sphere..).
- A Feature server, which stores extracted features from feature extraction algorithm such as MFCC, LPC, WLPC, using speeches from the data audio server
- A Mixture/distribution server, which stores speaker models computed from features located on the Feature server.
- A Statistic server, which contains the result of several algorithm computation(Average log-likelihood, Expectation-Maximization, MAP)

9.2 The LIA SpkDet toolkit

The LIA SpkDet toolkit aims at providing automatic speaker recognition tools using the ALIZE library, as shown in [16]. It is completed with other toolkit which allow to output figures, histograms,

visualizations useful for interpretation. The first tool, called *Energy Detector* aims at removing silences in speech data. The technology used consists in detecting the frame energy. Analysing all frame in an utterance provides an amplitude threshold value for which frame is considered a silence, or a speech frame. This method allows to filter unecessary information, and consequently saves non-negligible amount of time, as well as providing accuracy for model estimation and tests. *NormFeat* is meant to normalize frame distribution. *TrainTarget* produces speaker-specific gaussian mixture models. Subsequently, it contains feature extraction steps following normalization, in order to apply further modeling. *TrainWorld* performs universal background modeling on the set of training training features from all speakers. Both *TrainTarget* and *TrainWorld* use EM and MAP algorithms. *ComputeTest* provides the likelihood ratio score using a test segment, a model and a background model. The application is flexible and can receive multiple tests and enrollment speeches for modeling and statistical computation. It can be deployed to a use in a NIST evaluation campaign. The tools used in this Application benefits from the last state of the art advancements such as Support Vector machine [17], Factor Analysis, or Nuisance attribute projection (NAP) [9].

9.3 C++ code compilation

In order to use the LIA toolkit, one must first compile the ALIZE library. Once compiled, the executables are available. The user must modify several file accordingly in the "LIA SpkDet" folder to provide the right path to ALIZE library and executables. ALIZE library and LIA toolkit succeded for compilation with respective dependencies from LIA SpkDet and other optional toolkit towards ALIZE library.

9.3.1 LIA SpkDet modules

The LIA SpkDet toolkit provides several modules described in the previous section 9.2. Each of them holds a *test* folder which contains feature files. *TrainTarget* and *TrainWorld* can be launched and provide output results of GMMs with *means*, *covariances* and *mixture weights* values.

Further on, the goal was to use speech samples from the YOHO database to input them for enrollment. The YOHO database is composed with many speech files encoded in SPHERE format, which is a worldwide known basic soundfile format. The LIA SpkDet toolkit provides functions to read and handle SPHERE and raw data format.

Due to difficulties in running the application for SPHERE file format, the project has been abandoned after a long time, despite of the effort in studying the possible issues. Using MATLAB, a finalized program runs correctly. However it does not benefit of the latest state of the art methods due to several time and methodology constraints encountered along this project.

Part V

test data and evaluation

Contents

On previous chapters, we have described in detail a text-independent speaker identification system using gaussian mixture model. Additionally, we overlooked several speech processing algorithm used for different types of recognition systems such as: “text-dependent” speaker recognition, speaker verification. The next part aims at presenting the YOHO database and its previous uses in the field of speech processing, and then describing the results obtained from using YOHO database as a training and testing support for the system. The results will be presented analitically and using comparison tables.

Chapter 10

The YOHO speaker verification database

In the scope of training and testing speech samples on the speaker verification-identification system, the YOHO database provides a good quality recording platform.

It provides a large scale, high quality speech corpus to support text-dependent speaker authentication research, but it is equally useful for text-independent speaker recognition. The data was collected in 1989 by ITT under a US government contract but has not been abailable for public use before.

The YOHO database contains:

- "Combination lock" phrases (ex: 26-81-57). Each speaker provides a wide variety of phrases recording composed with a serie of 3 numbers. This is a simple technique which allows to record as many different sounds (vowels and consomns) as possible using only numbers.
- Collected over three-month period in a real world office environment. The length of the period has been chosen on purpose, to take into account the natural variation of the speaker's voice overtime.
- Four enrollment sessions per subject with 24 phrases per session. Each enrollment sessions last on a single day. The 4 sessions are distributed on a nominal time interval of three days between sessions.
- Ten test sessions per subject with four phrases per session. On the contrary to training sessions, where the number of recordings need to be high due to improve the background model training, testing sessions do not need to be long, as far as they provide sufficient features for speaker identification and taking into account real-life application where test utterances remain brief.
- 8kHz sampling with 3.8 kHz analog bandwidth.
- 1.5 gigabytes of data

The YOHO Speaker Verification Database is composed with exactly 108 male speakers and 30 female speakers. It was collected while testing a prototype speaker verification system by ITT Defense Communications Division under contract with the U.S Departement of Defense. The database is the largest supervised speaker verification database known to the authors.The number of trials and the number of test subjects were determined to allow testing at the 80 % confidence level to determine whether the system met specified performance requirements. The required error rates were 1% false rejection and 0.1 % false acceptance.

Chapter 11

Performance evaluation

This chapter aims at analysing the system's performance using output results. The first section deals with a reduced data and uses a likelihood table obtained while comparing the speakers to each model. The likelihood table is meant to emphasize the success of the speaker identification code.

11.1 Tests using a reduced YOHO data set

The actual section describes and outputs the performance of the system. First, using Gaussian mixture model, an example presenting the likelihoods obtained during a test on a sample of the YOHO database involving 4 speakers, randomly chosen. Using 24 training and test speech utterances per speaker. Second, an identical comparaison example using Universal background model

11.1.1 First method: Gaussian mixture model

speakers	s1	s2	s3	s4
s1	-9.4351	-10.1341	-10.3799	-12.2615
s2	-9.7287	-8.0112	-9.8564	-12.5933
s3	-10.4866	-10.3575	-7.9885	-10.6675
s4	-14.0281	-11.9361	-10.7364	-9.4907

Table 11.1: *Mean log-Likelihoods of each test speaker when compared to each training speaker using a simple setup of 12 Gaussians and 10 iterations*

The output results show that each unknown speaker, once compared with its respective training model, gives a higher mean likelihood (represented in red color), which confirms the identification of the unknown speaker.

Below, a second result table using different parameters:

speakers	s1	s2	s3	s4
s1	-9.4807	-10.8291	-11.1792	-12.0584
s2	-10.6503	-8.1751	-11.0263	-11.2049
s3	-12.1263	-11.2279	-8.5646	-11.8706
s4	-13.5519	-14.1931	-14.0690	-9.1627

Table 11.2: *Mean log-Likelihoods of each test speaker when compared to each training speaker using a simple setup of 64 Gaussians and 15 iterations*

11.1.2 Second method: Universal background model

The following tests indicate the performance output with UBM for each speaker.

speakers	s1	s2	s3	s4
s1	0.968	1.1727	1.1688	1.0221
s2	1.3373	0.9227	1.1012	0.9062
s3	1.1527	1.0494	0.773	1.2469
s4	1.2388	1.3496	1.3199	0.9657

Table 11.3: *likelihood ratio of each speaker*

The test succeeds for each of the 4 speakers. The next section deals with the entire YOHO data set and uses UBM further as a speaker "verification" technique. Consequently, a threshold value is chosen to either accept or reject a speaker.

11.2 Tests using the full YOHO speech data set

The following section outputs test results which include the entire YOHO speech database. The first method with Gaussian mixture model is a speaker identification technique, which only computes "false acceptance" as a performance indicator. The Universal background model calculates a likelihood ratio, see chapter 8. The tests allow to determine the appropriate threshold value. The ideal threshold is given with the EER. Obtaining the lowest equal rate implies finding a threshold value having a good tolerant/discriminative balance.

11.2.1 first method: Gaussian mixture model

A likelihood table is unnecessary, taking into account the amount of speakers involved. Hence, the system computes an error rate, as shown in [18], based on the number of false identification. The error rate is detailed more thoroughly in chapter 8. The following output result is obtained using several tests, each composed with different parameters. The parameters used for this purpose are:

- number of Gaussians
- number of iterations

Analysis parameters One speaker verification may analyse a given speaker's test speech and compares it with its training speech. On the basis of such principle, it can either accept or reject a test speaker for his/her claimed identity. Therefore, performance analysis for speaker verification is based on 3 parameters which are the "error rate", "false acceptance" and "false rejection".

Speaker identification does not reject a test speaker when calculating likelihoods between the test utterances and each training utterances. Therefore, performance analysis for speaker

H

false acceptance	number of speakers	Error rate
50	138	36.23 %
1	10	10 %
9	50	18 %
20	80	25 %
31	110	28.18 %

Table 11.4: *Performance analysis using different number of speakers*

identification is only based on "false acceptance", which corresponds to the number of errors, and allows us to compute the "error rate".

The "error rate" corresponds to the quotient of the number of false acceptance by the number of speakers, multiplicated by 100:

$$errorrate = \frac{numoffalseacceptance}{numofspeakers} * 100$$

Below, a graphical representation of the error Rate evolution according to the number of speakers:

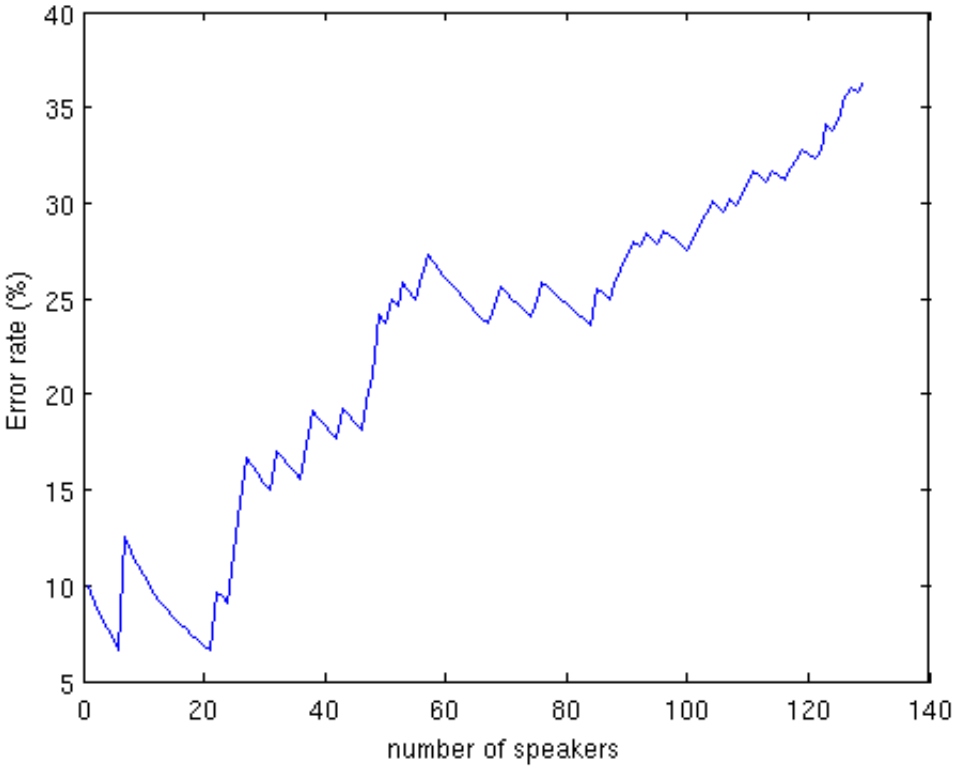

Figure 11.1: *Variation of the Error rate (%)*

H

false acceptance rate (FAR)	false rejection rate (FRR)	Threshold
0.02%	0.68%	0.60 %
0.14%	0.59%	0.65 %
0.95%	0.45%	0.70 %
3.54%	0.31%	0.75 %
10.0819%	0.199%	0.8 %
32%	0.06%	0.9 %
56.38%	0.026%	1.0 %

Table 11.5: *False acceptance rate (FAR), false rejection rate (FRR) and threshold value, [18]*

11.2.2 second method: Universal background model

Using universal background model, the results obtained for speaker identification are identical. Nevertheless, universal background model provides good result for speaker verification. Below, a representative table of *false acceptance* and *false rejection* rate according to the threshold value.

$$FAR = \frac{numberoffalseacceptance}{numberofspeakers^2} * 100$$

$$FRR = \frac{numberoffalserejection}{numberofspeakers^2} * 100$$

The given table clearly emphasizes the empirical nature of the threshold value's variation. Changing the threshold value influences the balance between FAR and FRR. Running a set of test computation while incrementing/decrementing the threshold value for each test will output the threshold generating the Equal error rate, hence best performances.

Below, a ROC curve, [18] which represents the false rejection as a function of false acceptance.

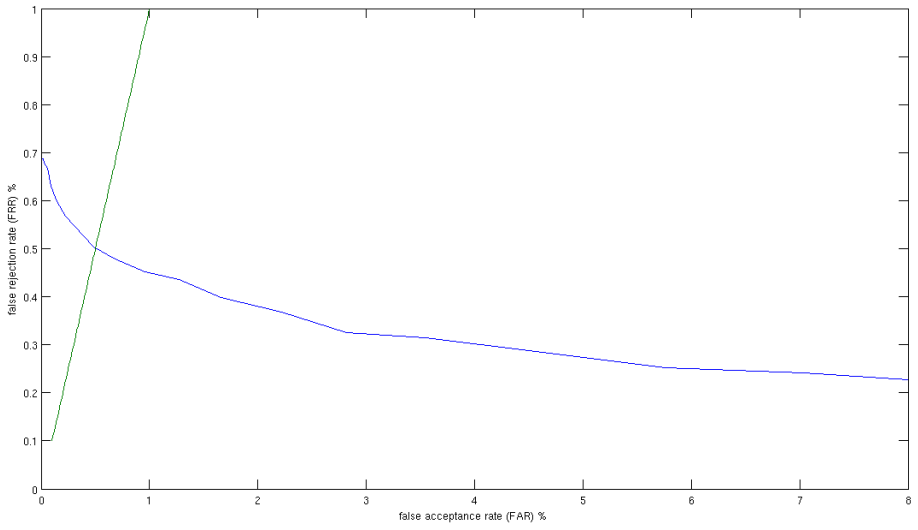

Figure 11.2: *ROC [18] curve between the false acceptance rate and the false rejection rate (Blue). A reference function $y = x$. Intersection = EER*

The intersection of the two curves gives the Equal error rate (EER). The plot displays an Equal error rate value for a false acceptance and rejection rounding around 0.5 . Having knowledge of this

value allow us to determine the threshold value corresponding. The threshold value corresponding to such EER is $\theta = 0.68$. We can safely assume the threshold value giving best performance for speaker verification is 0.68.

Chapter 12

Conclusion

Speaker recognition has encountered lots of advancements within the past few years, emerging new technologies improving robustness, more particularly in speaker verification. HMM provides interesting text-dependent speaker recognition, unlike the Gaussian mixture model which is fairly effective in the field of text-independent speaker recognition. GMM provides a robust basic model to compute likelihoods between a test speaker and a given model. This method has proven its effectiveness on small populations, with few noise components and intersession variability. Further, the universal background model brings the concept of a world model, created from all training speaker's features, using the EM algorithm. The UBM also saves computation time for the training of speaker-specific models. Using the MAP algorithm, it uses the new speaker's features to adapt the new background model, which is used as the speaker's model. The training phase is improved as well as the testing phase. Computation of the likelihood ratio makes it interesting for speaker verification making the match score range of different speakers comparable. The new techniques such as Support Vector Machine provide powerful discrimination to distinguish speaker and background. Nuisance attribute projection provides intersession variability and channel compensation, which partly eliminates dependencies towards recording quality and natural voice changes. State of the art for speaker recognition has improved significantly, which allows to perform various tests and performance analysis on the pool of existing technologies. The YOHO database provides a large data-set for experiments. Its use in the field of speaker recognition contributed to many advances, more particularly in speaker verification. This paper aimed to provide outputs from the use of YOHO database for GMM and UBM, as well as an overview among state of the art in speaker recognition. The results clearly demonstrates the improvements subject to UBM over simple GMM. The combination of both revealed significant result.
The latest technologies can significantly improve accuracy. Researchers study new techniques as well as improvement in the existing methods. Field of studies become larger, from physiological to behavioral with the use of high level features. The research in the field of speaker recognition contributes substantially to a better management in security for various use, although the behavioral aspects are only emerging recently.

Bibliography

[1] Minhdo, "Dsp mini project: An automatic speaker recognition system," http://www.ifp.illinois.edu/~minhdo/teaching/speaker_recognition/.

[2] M. J, "Linear prediction: A tutorial review," *Proceedings of the IEEE, IEEE Computer Society*, vol. 5, pp. 561–580, April 1975.

[3] H. Aki and L. Unto K, "A comparison of warped and conventional linear predictive coding."

[4] R. Douglas A and R. Richard C, "Robust text-independent speaker identification using gaussian mixture speaker models," *IEEE Transacations on Speech and Audio Processing*, vol. 3, january 1995.

[5] D. Reynolds, "Gaussian mixture models," pp. 659–663, 2009.

[6] R. Douglas, *Universal Background Models*, MIT Lincoln Laboratory 244 Wood St., Lexington, MA 02140, USA.

[7] K. Tomi and L. Haizhou, "An overview of text independent speaker recognition from features to supervectors," Master's thesis, University of Joensuu, Department of Computer Science and Statistics, Speech and Image Processing Unit, 2009.

[8] D. Reynolds, T. Quatieri, and R. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Process*, 2000.

[9] S. Alex, C. William M., and Q. Carl, "Nuisance attribute projection," mIT Lincoln Laboratory 244 Wood Street, Lexington, MA 02420.

[10] P. Fernando L and D. Jeffrey S, "Biometric authentication technology: From the movies to your desktop."

[11] M. H. Hayes, "Statistical digital signal processing and modeling," *John Wiley and Sons Inc.*, 1996.

[12] S. H. S. Salleh, A. Z. S. arneri, Z. Yusoff, S. A. R. A. Attas, L. S. Chieh, A. I. A. Rahman, and S. M. Tahir, "Speaker recognition based on hidden markov model," Digital Signal Processing Lab, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 2000.

[13] G. Michael, B. Ren, and P. Beat, "Quasi text-independent speaker-verification based on pattern matching," *Speech Processing Group, Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland*, 2007, iNTERSPEECH.

[14] U. Bulent, "Support vector machine," http://www.cac.science.ru.nl/people/ustun/.

[15] C. Eric, "Alize library user manual," 2008.

[16] M. Sylvain, M. Teva, L. Christophe, L. Anthony, C. Eric, B. Jean Francois, B. Laurent, F. Jerome, and R. Bertrand, "Plate forme open source d authentification biometrique," *JEP Avignon*, 2008.

[17] W. M. Campbell, J. P. Campbell, T. P. Gleason, D. A. Reynolds, and W. Shen, "Speaker verification using support vector machines and high-level features," *IEEE Transactions on Audio, Speech & Language Processing*, vol. 15, no. 7, pp. 2085–2094, 2007.

[18] J. Joseph P. Campbell, "Speaker recognition," *Department of Defense*.