Speaker Verification Systems under Various Noise and SNR Conditions

Qianhui Wan

A thesis submitted in partial fulfillment of
the requirements for the degree of Master of Applied Science in
Electrical and Computer Engineering

Ottawa-Carleton Institute for
Electrical and Computer Engineering
School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

In speaker verification, the mismatches between the training speech and the testing speech can greatly affect the robustness of classification algorithms, and the mismatches are mainly caused by the changes in the noise types and the signal to noise ratios. This thesis aims at finding the most robust classification methods under multi-noise and multiple signal to noise ratio conditions. Comparison of several well-known state of the art classification algorithms and features in speaker verification are made through examining the performance of small-set speaker verification system (e.g. voice lock for a family). The effect of the testing speech length is also examined. The i-vector/Probabilistic Linear Discriminant Analysis method with compensation strategies is shown to provide a stable performance for both previously seen and previously unseen noise scenarios, and a C++ implementation with online processing and multi-threading is developed for this approach.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Professor Martin Bouchard for providing invaluable guidance, thoughtful suggestions and prompt inspirations during my studies and research. His help and support consistently encouraged me to complete this work through the whole time. It is a great pleasure to have him as my supervisor and be a part of his research group.

I am thankful to my parents as well as my aunt for their support during my studies. Their love and understanding are invaluable and have supported me to complete this work.

I am grateful to my colleagues and friends Hilda and Hala, for their kind help and sharing. They have provided expert advice on both my research and my life. It is a great pleasure to work with them. Also, many thanks to my friends who have helped me through these two years.

Table of Contents

List of Figures

List of Tables

List of Acronyms

SNR	Signal to Noise Ratio
LPCC	Linear Predictive Cepstral Coefficients
PLP	Perceptual Linear Prediction
RASTA	RelAtive SpecTrAl
MFCC	Mel-Frequency Cepstral Coefficients
CMVN	Cepstral Mean Variance Normalization
GMM	Gaussian Mixture Model
UBM	Universal Background Model
SVM	Support Vector Machine
JFA	Joint Factor Analysis
PLDA	Probabilistic Linear Discriminant Analysis
SI-mPLDA	SNR-Independent Mixture of PLDA
SD-mPLDA	SNR-Dependent Mixture of PLDA
WCCN	Within-Class Covariance Normalization
LDA	Linear Discriminant Analysis
EER	Equal Error Rate
FAR	False Acceptance Rate
FRR	False Rejection Rate

Chapter 1 Introduction

1.1 Background

Speaker recognition is an important technique for biometric identification (Jain, Ross, & Prabhakar, 2004) which uses speech signal to identify human subjects. It consists of two main topics: speaker identification and speaker verification. Given a set of known speaker models and a new coming speech, speaker identification aims at finding which speaker does the speech belong to. In other words, speaker identification selects the speaker that sounds closest to the new speech. The task of speaker verification is to decide if the unknown speech belongs to the speaker it claims to be, which is achieved by comparing the similarity between the speech and a speaker model.

With the increasing popularity of portable devices such as cellphones, tablets and laptops, the industry of biometric authentication on these devices is growing significantly. Face recognition and fingerprint scan have already been applied to products for access purpose. Due to the ease of collection nowadays, speech information is considered to be the trend in industry (Hansen & Hasan, 2015) and this task can be accomplished by speaker verification.

Besides the application on identity authentication, speaker verification also has a vital role in forensics and law-enforcement purpose. In such cases, criminals may attempt to change their voices to avoid being recognized, and speaker verification techniques are able to make a reliable decision from the disguised speech recordings.

Based on the speech content constraints, speaker verification can be divided into two categories: text-dependent and text-independent. The former requires vocabularies from a certain group, such as digits zero to nine, while the other doesn't have any constraint on the speech content. Text-dependent speaker verification systems can achieve better accuracy compared to text-independent systems, since the vocabulary information can also be used for making the decision. Meanwhile the task for text-independent systems is more challenging without the constraints. In this case, only physiological features are available for verification and the system has to take into

account phonetic mismatch as well (Hansen & Hasan, 2015). This thesis focuses on text-independent speaker verification.

There are other challenges that are applicable to both text-dependent and text-independent systems, namely channel- and speaker-dependent variabilities. Channel-dependent variations, also known as "between-speaker" variations, contain the changes in recording devices, environmental noises and room reverberations; speaker-dependent or "within-speaker" variabilities are caused by the health condition, mood and changing age of speakers. These variations lead to mismatch between training data and testing data, resulting in verification accuracy degradation. A large amount of research has been conducted on compensation strategies to deal with those variabilities and great progress has been made. However, the mismatch condition of speech data still remains the most challenging problem in speaker verification, especially when the noise type and signal to noise ratio (SNR) conditions vary from speech to speech. Thus, this thesis aims at examining the performance of a small-set speaker verification system for different methods under multi-noise and multi-SNR conditions.

1.2 Literature Review

A speaker verification system consists of two main stages: feature extraction and classification. The former step collects the acoustic features from speech signal while the second step attempts to classify these features based on statistical strategies. Features such as prosody and conversation patterns are called "high-level" features, and these features are extracted from a relatively long sentence (greater than few seconds) (Fazel & Chakrabartty, 2011) and require high-complexity computation for processing. However, considering the complexity constraint for real-life applications as well as the limitation of data amount, most speaker verification systems use only "low-level" features like Linear Predictive Cepstral Coefficients (LPCC) (Makhoul, 1975) and Mel Frequency Cepstral Coefficients (MFCC) (Davis & Mermelstein, 1980) in practice. Unlike the "high-level" ones, these features can be extracted from a short-time window which is normally around 25 milliseconds (interval for speech stationarity). There is no exact conclusion on which features give better results, since it normally depends on the system setting such as data conditions and classification strategies.

The main idea behind LPCC is based on Linear Prediction Coding (LPC) which assumes that the current speech sample can be predicted from the past samples. An all-pole filter is used for modeling the vocal tract in LPCC and the parameters of the filter are estimated through auto-regression. Finally the LPCCs are computed from the prediction coefficients by using a recursive method. There is another related feature extraction approach called Perceptual Linear Prediction (PLP) (H. Hermansky & Cox, 1991) that comes from LPC and which also exploits the psycho-acoustic properties of human ears.

MFCCs are the most commonly-used features in speaker verification systems. A set of triangle Mel filterbank is used for converting the spectrum into Mel scale. These filterbank gives the energy information of the speech signal contained within a certain frequency range. More specifically, the Mel filterbank puts more focus on the low frequency part, where major information of speech is located. Therefore, MFCCs are widely used in speech recognition and speaker recognition. Since features like LPCCs and MFCCs are computed from short-length frames, they lack the dynamic information in speech signals. It has been proved that the system performance can be improved through appending the first- and second-order derivatives to the original coefficients (Furui, 1986).

There is an additional pre-processing approach called Voice Activity Detection (VAD) that is sometimes performed on features before sending them to training and verification phases. VAD aims at finding only the features that are extracted from voiced frames and discarding the ones from non-voiced part. This can be achieved by estimating the Signal to Noise Ratio from frame to frame and only keeping the high-SNR ones. However, since speech signals are most likely collected from different environments and noise conditions, this task may be very challenging.

The second stage of a speaker verification system is segregating the features of one speaker from others given some enrollment speech. In other words, speaker models are trained according to each involved speaker, which can represent the identity of each speaker. During the verification phase, an unknown speech is compared with a speaker model to decide the similarity between them. The modeling methods can be categorised into two types: generative methods and discriminative methods. The former one uses only the speech data from the target speaker to train

the speaker model, these methods are able to capture the specific statistical properties of a speaker. Meanwhile discriminative methods adopt not only speech from the target speaker but also imposters, and the core is distinguishing the statistics of the target speaker from the imposters.

A classic representative of generative methods is Gaussian Mixture Model (GMM) (D. A. Reynolds & Rose, 1995) and a lot of expansions have been developed based on it. The core idea behind GMM is the assumption that features follow the Gaussian distribution, and each speaker is represented by a GMM. The final decision at the verification phase is based on the similarity between one speaker GMM and an unknown speech. GMM is an effective method for speaker identification, however, speaker verification may also deal with imposters. Hence, the Universal Background Model (UBM) is proposed to improve the performance for open-set speaker verification systems (Douglas A Reynolds, Quatieri, & Dunn, 2000). Speaker-based GMMs are derived, or adapted, from a pre-trained UBM which contains information about a large amount of imposters.

One of the challenges of speaker verification is that many classification methods are based on fixed-length features meanwhile the duration of the involved speech varies. Therefore, effort has been put to find a fixed-dimension feature, and GMM based supervector has been proposed (Kenny, Mihoubi, & Dumouchel, n.d.). A GMM supervector is calculated by concatenating the GMM mean vectors of an adapted speaker model. And since the mixture number for each adapted-GMM is the same for each speaker, the resulting supervectors are of the same dimension regardless of how long the actual speech is. Hence, this approach is normally followed by Support Vector Machines (SVM) which require fixed-dimensional inputs.

SVM is a typical discriminative classification method that aims at finding an optimal hyperplane that can separate the data from two different classes (Cortes & Vapnik, 1995). Research has been done to prove that GMM supervectors/SVM is an effective way for speaker verification (Campbell, Sturim, & Reynolds, 2006). The fixed dimension supervectors extracted from one speaker model are treated as a positive (true) input of the SVM while supervectors from imposters serve as the negative (false) input. However, in real life the amount of speech data of one target speaker may be limited, therefore there can be an imbalance between positive and negative data.

Though this problem can be addressed at some level by duplicating the target data, the system performance would still be affected. Meanwhile, due to the demand of robustness under training data and testing data mismatch, researchers kept finding new approaches that do not require information from imposters and which are also robust to variabilities.

In (Kenny, 2005), Kenny proposed a Joint Factor Analysis (JFA) model that assumes GMM supervectors which contain both speaker and channel variations. In JFA, each GMM-supervector is modeled by two factors: speaker factor and session factor. In other words, it assumes that a supervector can be decomposed to three parts: a speaker- and channel- independent component, a speaker-dependent part and the other one only depends on channel effects. This approach outperformed other approaches since it takes into consideration both within- and between-speaker variabilities.

A novel approach called i-vector was developed in 2011 (N. Dehak, Kenny, Dehak, Dumouchel, & Ouellet, 2011) and remains the state-of-art so far. Unlike in JFA where a speaker supervector is divided into a speaker subspace and a channel subspace, i-vector defines only one total variability space to model the speaker and session variability. Each utterance is represented by a fixed-dimensional i-vector, which can be seen as a "feature" of each sentence. The combination of i-vector and SVM appears to be robust under mismatch situations (N. Dehak et al., 2011). Two other scoring methods were also proposed, a cosine distance scoring and a Probabilistic Linear Discriminant Analysis (PLDA) scoring method. The former one measures the cosine distance between two i-vectors and it is employed for its simplicity. PLDA models the speaker and channel information contained in an i-vector, like in JFA. It assumes that each i-vector follows a single Gaussian distribution, and can be decomposed into a speaker-dependent part and a speaker-independent part. I-vector/PLDA remains the most successful approach in speaker verification (Kenny, 2010).

The robustness under noisy conditions has become the focal point in speaker verification since 2012. With the success of i-vector/PLDA approach, the idea of Mixture of PLDA was first proposed in (M.-W. Mak, 2014) and further formalized in (M. W. Mak, Pang, & Chien, 2016a). It was argued that SNR makes i-vectors shift in i-vector space, thus each PLDA model should only

be employed to describe i-vectors from a certain SNR range. This assumption led to two new approaches: SNR-Independent Mixture of PLDA (SI-mPLDA) and SNR-Dependent Mixture of PLDA (SD-mPLDA). Both use a mixture of Gaussian to describe an i-vector, while the latter one introduces an additional SNR indicator that guides the clustering. Both Mixture of PLDA methods showed better performance than conventional Gaussian PLDA, with a price of increasing computational complexity.

Despite improving the robustness of modeling methods themselves, there are also compensation strategies that can be performed at both the feature extraction stage and classification stage. Approaches such as cepstral mean subtraction (Furui, 1981) and RASTA filtering (H. Hermansky & Morgan, 1994) can be applied to raw features before classification. Meanwhile linear discriminant analysis (LDA) (Bishop, 2006), within-class covariance normalization (WCCN) (Hatch, Kajarekar, & Stolcke, 2006) and i-vector length normalization (Romero & Wilson, 2011) can be performed before the modeling stage.

As for the system performance under mismatch conditions, which we are interested in, there are two common approaches to address the accuracy degradation: multi-condition training where speech under different conditions are pooled together for building one speaker model, and multiple condition-dependent speaker models training. Due to the great success of the i-vector/PLDA approach, most studies on mismatch conditions are based on this setup. In (Lei, Burget, Ferrer, Graciarena, & Scheffer, 2012), different cocktail noises are added to clean speech at 20, 15 and 8 dB SNR and this is applied to both enrollment and testing data. It was found that the Equal Error Rate (EER) on noisy cases drops from 4 to 13 times compared to the clean setup. However, this degradation can be reduced by nearly 40% by adding noise to PLDA training data.

Similarly, (Rajan, Kinnunen, & Hautamaki, 2013) examined the effect of multi-condition training and score averaging under noisy situations. Normally one speaker may have more than one enrollment speech, therefore there would be several enrollment i-vectors for each speaker. There are two common strategies for scoring in this case: 1) averaging i-vectors before scoring and make one final score based on the averaged i-vector, 2) making scores based on all existing i-vectors and averaging the scores. During the experiments four noises were added to clean speech

at 15 and 6 dB SNR, and the most robust results under noisy testing conditions were obtained when multi-condition training was applied to both enrollment and PLDA training. With respect to scoring methods, averaging i-vectors before scoring outperformed score averaging.

Experiments were conducted to evaluate the effectiveness of multiple model training in (Garcia-Romero, Zhou, & Espy-Wilson, 2012), where several PLDA models were trained with speech at different SNR ranges. Enrollment and PLDA training speech were added with different noise at 20 dB, 10 dB, 6 dB and 0 dB SNR respectively. This results in 16 replicas of original training data and all of them are used for building a multi-condition model. It was found that multiple model training achieved similar results as multi-condition training where all speech are pooled together, although the latter one requires much less computation complexity.

In (M.-W. Mak, 2014), PLDA training data were combined with noise at 15 and 6 dB, and the noisy data along with original speech were used for conventional PLDA and SD-mPLDA training. The same noise-adding strategy was applied to enrollment speech as well. It was shown that SD-mPLDA was more robust than traditional PLDA under mismatch conditions. The idea of mPLDA was further developed in (M. W. Mak et al., 2016a), where noise were added to PLDA training data for nearly uniformly distributed SNRs from 0 to 20 dB. Mixture of PLDA was shown to be more robust than conventional PLDA when noise is added to training data at 15 dB and 6 dB SNR.

1.3 Contributions

In previous studies, the mismatch condition generation was normally achieved by adding noise to clean speech at several specific SNRs. Even though PLDA training data have a nearly flat SNR distribution in (M. W. Mak et al., 2016a), the enrollment speech still only has 3 different SNR which are the same as the testing speech. However, in practice speech is not likely to fit in a few specific SNR values and noise types, therefore this thesis evaluates the performance of different approaches when the training speech is randomly combined with different noise at different SNR. Moreover, inspired by (Pang & Mak, 2015) where linear fusion of SNR-Independent PLDA and SNR-Dependent PLDA produced an improvement compared with other

methods, linear fusion of scores is also applied to several approaches that previously produced good performance.

Additionally, this thesis investigates the shortest speech length that is required to obtain acceptable results under this multi-condition situation. For real-life speaker verification applications, the task is to make a verification decision based on a quite short speech. It may not be pleasant for speakers to speak for more than 10 seconds to get a valid decision. Hence, it is necessary to evaluate the effect of testing speech length on the verification performance, meanwhile finding the shortest acceptable speech length under real-life mismatch conditions.

The above contributions have led to the following publication:

Q. Wan, M. Bouchard "Performance Evaluation of Mixtures of PLDA and Conventional PLDA for a Small-Set Speaker Verification System", *Proc. 30th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2017)*, Windsor, Ontario, April 30 - May 3, 2017

To achieve the objectives of this thesis, experiments were first conducted on MATLAB R2015b. The MATLAB code is based on VOICEBOX (VOICEBOX: Speech Processing Toolbox for MATLAB, n.d.), MSR Speaker Recognition Toolkit (Sadjadi, Slaney, & Heck, 2013), JFA cookbook (Joint Factor Analysis Matlab Demo | Speech Processing Group, n.d.) and mPLDA code package from The Hong Kong Polytechnic University (Mixture of PLDA, n.d.). VOICEBOX was used for feature extraction, noisy speech generation and SNR measurement, while the other toolkits were used for classification.

Based on the experiments performed using MATLAB, considering both the robustness and the computational complexity of algorithms, an online-processing system was then implemented in C++. This implementation was completed based on the Kaldi toolkit (Kaldi ASR, n.d.). The resulting real-time system is to be integrated into the product of a smart-camera video surveillance local company, which has sponsored this research. This is another contribution of this thesis.

1.4 Organization

This thesis is organized as follow:

Chapter 2 gives the descriptions of three commonly used features in speaker verification: linear prediction cepstral coefficients, perceptual linear prediction coefficients and Mel-frequency cepstral coefficients. A feature compensation strategy, cepstral mean variance normalization, is also introduced.

Chapter 3 provides detailed explanations of the classification methods, from the most basic Gaussian Mixture Model to the state-of-art i-vector and Probability Linear Discriminant Analysis (PLDA), as well as two new extensions of PLDA proposed in 2016.

Chapter 4 introduces three compensation strategies that are normally applied after the i-vector approach: i-vector length normalization, within-class covariance normalization and linear discriminant analysis.

Chapter 5 evaluates the performance of the features and algorithms described in Chapter 2 and 3, under both clean and multi-noise multi-SNR conditions. Compensation strategies are also examined as well as three linear fusion systems. Moreover, the effect of the testing speech length on the performance is also tested in chapter 5.

Chapter 6 describes aspects of a C++ implementation of the i-vector/PLDA approach, with performance of continuous online processing.

Chapter 7 presents the conclusions and discussions of this thesis.

Chapter 2 Feature Extraction

2.1 Linear Prediction Cepstral Coefficients

This chapter first gives a brief introduction of Cepstral Coefficients. Given a set of time series $x(n)$, the corresponding frequency spectrum $x(k)$ and estimate of the power spectrum $P(k)$ can be obtained through a Discrete Fourier Transform:

$$x(k) = \mathrm{DFT}(x(n)) \tag{2.1.1}$$

$$P(k) = |x(k)|^2 \tag{2.1.2}$$

Note that to have true PSD units, the PSD estimate $P(k)$ needs to be divided by the signal sample length, but this is omitted here because it is just a scaling factor that does not affect classification results. Autocorrelation is a widely used technique in signal processing for analyzing series, it can be computed from a given $P(k)$, assuming that proper zero padding was applied to compute $x(k)$:

$$A(n) = \mathrm{IDFT}(P(k)) \tag{2.1.3}$$

The calculation of the cepstrum is similar to the autocorrelation $A(n)$ with an additional logarithm operation before the IDFT. Note that the "real" cepstrum is used here, rather than the "complex" cepstrum, as the $C(n)$ coefficients are real-valued from the even symmetry of the real-valued (and positive) $P(k)$:

$$C(n) = \mathrm{IDFT}(\log(P(k))) \tag{2.1.4}$$

Thus, the cepstrum can be seen as a log compressed version of the autocorrelation, and the logarithm allows us to apply cepstral mean subtraction which will be described later on. LPCC is

computed the same way as the standard cepstral coefficients except that it uses an auto-regressive power spectrum estimate instead of $P(k)$ (Makhoul, 1975). A linear prediction model can be represented as:

$$y(n) = -\sum_{k=1}^{K} a(k)\, y(n-k) + b(0)\, w(n) \tag{2.1.5}$$

where the current signal is predicted from past samples, and $w(n)$ is white noise with zero mean and variance σ_w^2. The equation becomes an auto-regression or all-pole model in the *z*-domain:

$$Y(z) = H(z) W(z) \tag{2.1.6}$$

with

$$H(z) = \frac{b(0)}{1 + a(1) z^{-1} + a(2) z^{-2} + \cdots + a(K) z^{-K}} \tag{2.1.7}$$

where a and b are called prediction coefficients which can be estimated through Yule-Walker equations (Jackson, 1996). Based on this equation, the power spectrum estimate in the frequency domain can be computed as:

$$S_{AR}(f) = \frac{\sigma_w^2}{\left| 1 + \sum_{m=1}^{K} a(m) e^{-j m 2\pi f T} \right|^2} \tag{2.1.8}$$

where the denominator is equivalent to $\left|\mathrm{DFT}([1, a(m)])\right|^2$ with zero padding and some frequency scaling. The LPCCs are then computed through this auto-regression power spectrum estimate:

$$LPCC = \mathrm{IDFT}\left(\log\left(S_{AR}(f)\right)\right) \tag{2.1.9}$$

2.2 Perceptual Linear Prediction

PLP is an extension of LPCC, where the power spectrum estimate $P(k)$ is converted to a critical-band spectrum first before auto-regressive modelling. It came with the idea that conventional LPCC puts an even attention to all frequency parts, but the spectral resolution of human hearing drops beyond 800Hz (Hermansky, 1990). Generally speaking, PLP introduces three perceptual aspects in analysis: critical-band analysis, equal loudness curve and the intensity-loudness power-law relation. Critical band filters model the "auditory filter" in the human cochlea, thus it is believed that they provide more specific speech information. The additional steps in PLP extraction can be summarized as:

1. $P(k)$ is first warped to a bark-scale through a hertz -> bark transformation. The bark-scale is a nonlinear frequency scale of hearing that compresses the upper end of $P(k)$ (Hermansky, 1990).
2. Convolving $P(k)$ with the spectrum of a simulated critical-band masking curve, resampled in 1-Bark intervals.
3. Pre-emphasize the result of the convolution by the simulated equal-loudness curve as in Equation (2.2.1). This is due to the different sensitivity of human hearing at different frequencies, and $E(\omega)$ here is an approximation:

$$E(\omega) = \frac{\left[\left(\omega^2 + 56.8\times10^6\right)\omega^4\right]}{\left[\left(\omega^2 + 6.3\times10^6\right)^2\left(\omega^2 + 0.38\times10^9\right)\right]} \qquad (2.2.1).$$

4. The intensity-loudness power law is also known to be a cubic-root amplitude compression, it is used to model the nonlinear relation between the intensity of sound and its perceived loudness (Hermansky, 1990).

After these four steps, PLP computation follows the same process as LPCC computation. Rasta-PLP is a popular extension of PLP, which suppresses the sensitivity of spectral analysis to

slow variations in the spectrum (coming from changes in the recording environments). This is achieved by applying a spectral subtraction to the critical-band power spectrum before the equal-loudness operation.

2.3 Mel-Frequency Cepstral Coefficients

MFCCs are a widely used feature in the Automatic Speech Recognition and Speaker Recognition fields. The shape of the vocal tract, which contains the phoneme information of speech, can be represented by the envelope of the short-time power spectrum, and MFCCs are able to represent this envelope accurately. The whole extraction process contains the following six stages:

1. Divide speech data into overlapped short frames: Since an audio signal changes through time, it is better to analyze it by segmenting speech into short frames where each segment can be seen as statistically stationary. In most work, a speech signal is framed into 25 ms with 15 ms overlap.
2. Calculate the periodogram of each frame: Given a single frame in the time domain $x_i(n)$ where i is the frame number, the corresponding samples in the frequency domain $X_i(k)$ are computed by first applying a Hamming window with length N, then applying a DFT to the windowed frame. The periodogram estimate of $x_i(n)$ is computed as:

$$P_i(k) = \frac{\left|X_i(k)\right|^2}{N} \tag{2.3.1}$$

   As previously mentioned, the factor $\frac{1}{N}$ can be omitted in the considered application.

3. Apply Mel filterbank to $P_i(k)$ then sum up the energies in each filter: Mel filterbank is a set of triangle filters at Mel-scale (Stevens, Volkmann, & Newman, 1937). The transformation between Mel-scale and Hertz-scale is:

$$M(f) = 1125 \ln\left(1 + \frac{f}{700}\right) \tag{2.3.2}$$

$$M^{-1}(m) = 700\left(\exp\left(\frac{m}{1125}\right) - 1\right) \tag{2.3.3}$$

where f and m are frequencies in Hertz and Mel scale respectively. Figure 2-1 describes this relationship in more details. It is notable that the Mel-frequency scale cares more about the low-frequency part than the high-frequency part (in Hertz). This is inspired by the human hearing mechanism where it is easier for humans to tell the difference between two low frequencies (i.e. 100 Hz and 200 Hz) than two high frequencies (i.e. 10100 Hz and 10200 Hz). An example of an ideal Mel filterbank of 26 filters is presented in Figure 2-2, this is a typical filterbank for 8000 Hz sampled signals with a 300 Hz (401.97 Mels) lower frequency and an upper frequency equals to 3700 Hz (2071.73 Mels). The indexes of each triangle filter is determined by finding 26 uniformly distributed frequencies within 401.97 to 2071.73 Mels. In some applications where mapping human perception is key, the amplitudes of the filters also decrease as the central frequency increases, which aims at putting more weight on the low frequency part. In this thesis and as in previous related work in speaker verification, the same weight/amplitude is used for all filters. After applying the filterbank to the periodogram estimation of each frame, the energies from each filter are summed up, therefore 26 energies are computed using the above filterbanks.

4. Compute the log of each energy, followed by applying a DCT to the corresponding log energies, the resulting coefficients are the MFCCs. In speech/speaker recognition, only the lower 13 or 19 coefficients are normally kept. Figure 2.3 shows the complete extraction process over a frame of 200 samples.

Since MFCCs contain only the information of short periods, “velocities” and “accelerations” computed over multiple frames can be appended to the original MFCCs. These dynamic features are called delta and delta-delta features. Delta features can be calculated by:

$$delta_i = \frac{\sum_{n=1}^{N} n\left(c_{i+n} - c_{i-n}\right)}{2\sum_{n=1}^{N} n^2} \tag{2.3.4}$$

where i is the frame number, c are MFCCs and N is typically set to four. Delta-delta features are computed from delta features instead of MFCCs through the same way.

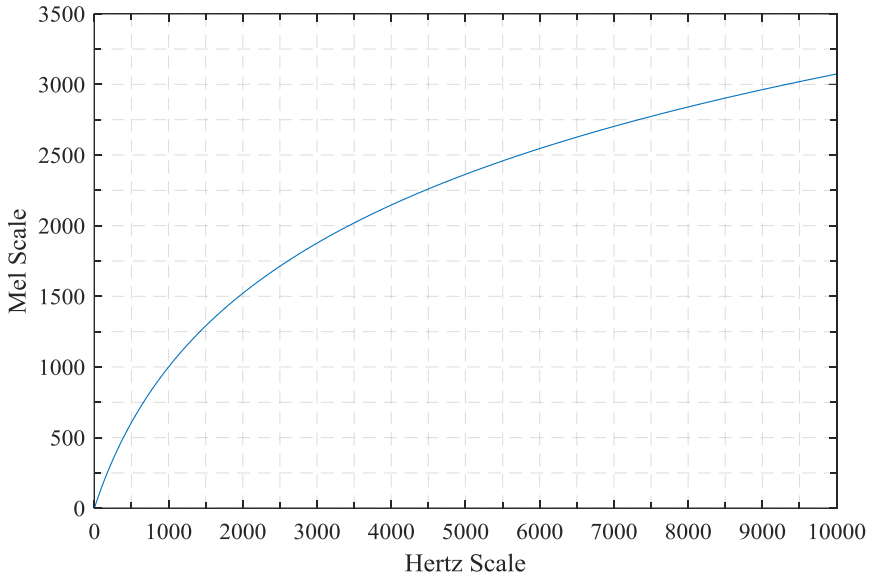

Figure 2-1: Mel-Hertz transform

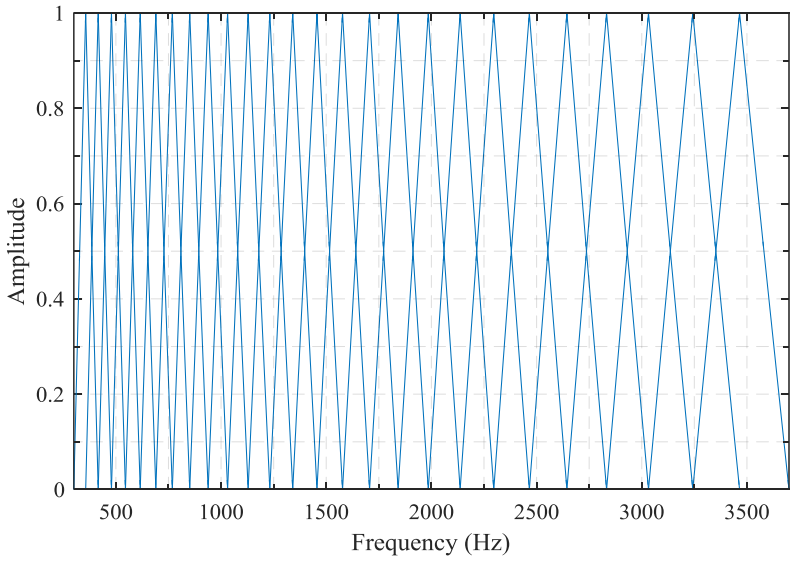

Figure 2-2: An example of a Mel filterbank of 26 filters

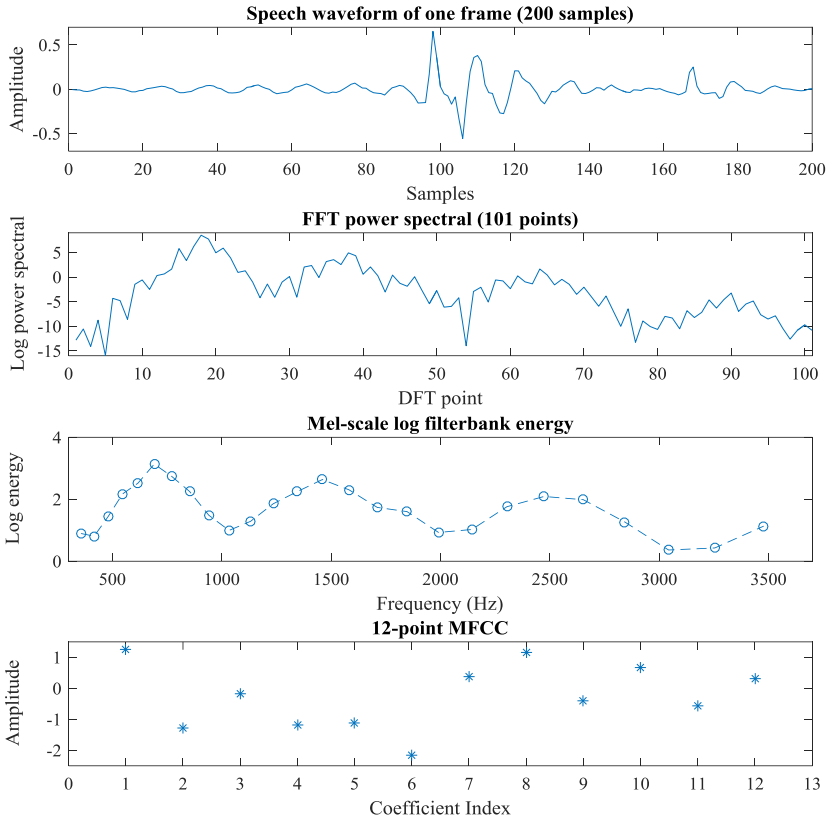

Figure 2-3: 12-order MFCCs extraction process over a frame of 200 samples

2.4 Cepstral Mean Variance Normalization

Cepstral Mean Variance Normalization (CMVN) is a widely used technique for addressing the mismatch between enrollment data and testing data. In real life, a recorded speech $c(n)$ is the convolution of an "actual" speech $s(n)$ and a channel impulse response $h(n)$ from the environment:

$$c(n) = s(n) * h(n) \tag{2.4.1}$$

According to the steps of extracting cepstral coefficients, the resulting cepstrum after FFT and log computation is:

$$C(f) = S(f) \bullet H(f) \tag{2.4.2}$$

$$C(q) = IDFT(\log[S(f) \bullet H(f)]) = S(q) + H(q) \tag{2.4.3}$$

The convolution becomes a simple addition in the cepstral domain and CMVN is based on this knowledge. Assume that $C(m,k)$ is the k th mel-cepstral coefficient from the m th frame, the normalized feature after applying CMVN is computed as:

$$\tilde{C}(m,k) = \frac{C(m,k) - \mu(m,k)}{\sigma(m,k)} \tag{2.4.4}$$

where $\mu(m,k)$ is the mean while $\sigma(m,k)$ refers to the standard deviation calculated by:

$$\mu(m,k) = \frac{1}{N} \sum_{i=m-\frac{N}{2}}^{m+\frac{N}{2}} C(i,k) \tag{2.4.5}$$

$$\sigma^2(m,k) = \frac{1}{N} \sum_{i=m-\frac{N}{2}}^{m+\frac{N}{2}} \left(C(i,k) - \mu(m,k)\right)^2 \tag{2.4.6}$$

where N is the length of a sliding window which centers on the current frame. To simplify the analysis, Equation (2.4.5) can also be written as:

$$\mu(q) = \frac{1}{N} \sum_j C_j(q) \tag{2.4.7}$$

where j is the frame number. By substituting (2.4.7) to (2.4.3), the mean becomes:

$$\mu(q) = H(q) + \frac{1}{N} \sum_j S_j(q) \tag{2.4.8}$$

Then subtracting $\mu(q)$ from $C(q)$ gives:

$$C(q)-\mu(q)=\left[S(q)+H(q)\right]-\left[H(q)+\frac{1}{N}\sum_{j}S_j(q)\right]$$
$$C(q)-\mu(q)=S(q)-\frac{1}{N}\sum_{j}S_j(q) \qquad (2.4.9).$$

It can be noticed that there is no channel response in Equation (2.4.9). The normalized features have zero mean and unit variance in each feature dimension.

Chapter 3 Classification

3.1 Gaussian Mixture Models

The Gaussian or Normal Distribution is a common probability distribution and is widely used in statistics. Given a set of F -dimensional random variables $\mathbf{X}=\left(x_1, x_2, \ldots, x_F\right)^{\mathrm{T}}$, the Gaussian distribution of $\mathbf{X}$ is defined as:

$$N\left(x \mid \mu, \Sigma\right)=\frac{1}{\left(2\pi\right)^{N/2}\sqrt{|\boldsymbol{\Sigma}|}}\exp\left(-\frac{1}{2}\left(x-\mu\right)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\left(x-\mu\right)\right) \tag{3.1.1}$$

where $\mu=E[\mathbf{X}]$ is the mean and $\boldsymbol{\Sigma}=E[(\mathbf{X}-\mu)(\mathbf{X}-\mu)^T]$ is the covariance matrix. However, a single Gaussian distribution may not be enough to describe the distribution of a set of data in reality. Hence Gaussian Mixture Models are introduced to deal with more complicated distributions. A GMM can be seen as the combination of single Gaussian distributions, and the probability of a C -mixture GMM is defined as:

$$p\left(x\right)=\sum_{c=1}^{C} w_c \cdot N\left(x \mid \mu_c, \boldsymbol{\Sigma}_c\right) \tag{3.1.2}$$

where w_c is called the weight of the c^{th} Gaussian and $\sum_{c=1}^{C} w_c = 1 \quad 0 \le w_c \le 1$. A simple illustration of a GMM is described in Figure 3-1, which presents a 1-dim case where three Gaussians share the same weight, as well as a $2D$ case. The task of GMM modeling in speaker verification is as follows: given a set of acoustic features $\mathbf{X}=\left(x_1, x_2, \ldots, x_F\right)$ which is assumed to follow a Gaussian mixture distribution, estimate the parameter set $\theta=\left\{w_c, \mu_c, \boldsymbol{\Sigma}_c,\ c=1,\ldots,C\right\}$ of the distribution that fits the data. Assuming that the data points in $\mathbf{X}$ are independent, the solution to this task is to find the theta that maximizes the likelihood $p\left(\mathbf{X} \mid \theta\right)$:

$$\theta_{opt} = \arg\max_{\theta} p(\mathbf{X}|\theta) = \arg\max_{\theta} \prod_{i=1}^{F} p(x_i|\theta) \tag{3.1.3}$$

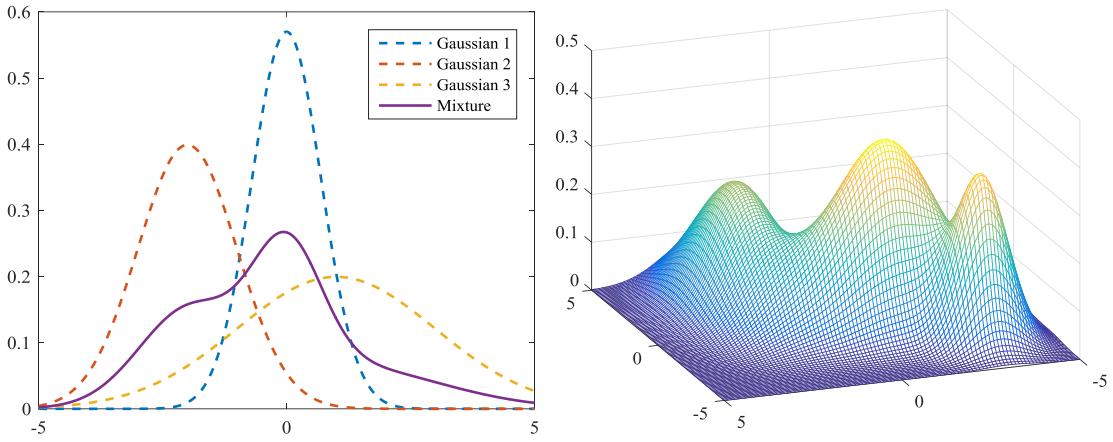

Figure 3-1: 1D Gaussian distribution (left) and 2D Gaussian distribution (right).

where θ_{opt} is the optimal parameter set, each x_i is a feature vector of dimension d (e.g. a 39-dim MFCC vector). This maximization process can be accomplished through the well-known Expectation Maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977). A hidden variable Z (a.k.a. latent variable) is introduced to make the process simpler. According to EM, maximization of the likelihood with the form $p(\mathbf{X}|\theta)$ can be solved by maximizing of the function $Q(\theta, \theta^{\mathrm{T}})$:

$$Q(\theta, \theta^{\mathrm{T}}) = E_Z\left[\log\left(p(\mathbf{X}, Z|\theta^{\mathrm{T}})\right)\right] \tag{3.1.4}$$

where θ^{T} stands for the "previous" parameter set and $E_Z[\ ]$ is the expectation computation regarding Z. Equation (3.1.2) for a single data point x_i from $\mathbf{X}$ is:

$$p(x_i) = \sum_{c=1}^{C} w_c \cdot N(x_i|\mu_c, \mathbf{\Sigma}_c) \tag{3.1.5}$$

The joint likelihood of $\mathbf{X}$ and Z is:

$$p(\mathbf{X}, Z \mid \theta) = \prod_{i=1}^{F} \prod_{c=1}^{C} \left[p(x_i, c \mid \theta) \right]^{z_{ic}}$$
$$p(\mathbf{X}, Z \mid \theta) = \prod_{i=1}^{F} \prod_{c=1}^{C} \left[p(x_i \mid c, \theta) \right]^{z_{ic}} \left[p(j \mid \theta) \right]^{z_{ic}} \tag{3.1.6}$$

where z_{ic} represents an indicator variable:

$$z_{ic} = \begin{cases} 1 & \text{if } x_i \text{ follows the } c^{\text{th}} \text{ Gaussian} \\ 0 & \text{otherwise} \end{cases} \tag{3.1.7}$$

Substituting (3.1.6) to (3.1.4) gives:

$$Q(\theta, \theta^{\mathrm{T}}) = \sum_{i=1}^{F} \sum_{c=1}^{C} E_Z \left(z_{ic} \mid \theta^{\mathrm{T}} \right) \log \left[p(x_i \mid c, \theta) \right] + E_Z \left(z_{ic} \mid \theta^{\mathrm{T}} \right) \log \left[p(c \mid \theta) \right] \tag{3.1.8}$$

$$E_Z \left(z_{ic} \mid \theta^{\mathrm{T}} \right) = \frac{p(c, \theta^{\mathrm{T}}) \, p(x_i \mid c, \theta^{\mathrm{T}})}{p(x_i, \theta^{\mathrm{T}})} \tag{3.1.9}$$

The maximization is computed by $\dfrac{\partial Q(\theta, \theta^{\mathrm{T}})}{\partial \theta} = 0$ and it gives:

$$\tilde{\mu}_c = \frac{\sum_{i=1}^{F} p(c \mid x_i, \theta^{\mathrm{T}}) x_i}{\sum_{i=1}^{F} p(c \mid x_i, \theta^{\mathrm{T}})} \tag{3.1.10}$$

$$\tilde{\Sigma}_c = \frac{\sum_{i=1}^{F} p(c \mid x_i, \theta^{\mathrm{T}}) \left[(x_i - \mu_c)(x_i - \mu_c)^{\mathrm{T}} \right]}{\sum_{i=1}^{F} p(c \mid x_i, \theta^{\mathrm{T}})} \tag{3.1.11}$$

w_c can be estimated through introducing a Lagrange multiplier that leads to:

$$\tilde{w}_c = \frac{1}{F}\sum_{i=1}^{F} p\left(c \mid x_c, \theta^{\mathrm{T}}\right) \tag{3.1.12}$$

where

$$p\left(c \mid x_i, \theta^{\mathrm{T}}\right) = \frac{w_c N\left(x_i \mid \mu_c, \boldsymbol{\Sigma}_c\right)}{\sum_{k=1}^{C} w_k N\left(x_i \mid \mu_k, \boldsymbol{\Sigma}_k\right)} \tag{3.1.13}$$

In speaker verification, each speaker is presented by one GMM trained on some speech data from that speaker. Given a GMM speaker model $\mathbf{GMM_s}$ from speaker s and a testing sequence $\mathbf{X}$ extracted from a testing speech, the probability that the testing speech belongs to s is calculated through the log likelihood:

$$\mathrm{LLK}\left(\mathbf{GMM_s}, \mathbf{X}\right) = \log p\left(\mathbf{X} \mid \theta_s\right) \tag{3.1.14}$$

The GMM modeling in speaker verification can be summarized in Table 3-1.

Table 3-1: Modeling process of GMM

Model training:
Given a set of training features of speaker s : $\mathbf{X} = (x_1, x_2, \ldots, x_F)$, initialize $p(x)$ with random θ using (3.1.1) & (3.1.2).
Iterate and update θ using (3.1.10), (3.1.11) and (3.1.12).
Testing:
Given a model GMM_s and an observation X , calculate the likelihood according to (3.1.14).
Make a decision based on the likelihood.

3.2 Universal Background Models

GMM-based Universal Background Models (UBM) are built on the goal of maximizing the distance between a target speaker and impostors. Research has shown that the optimal way to

compute the likelihood ratio between a speaker model $\mathbf{GMM_s}$ and an observation $\mathbf{X}$ is (Douglas A Reynolds et al., 2000):

$$\frac{p(\mathbf{X}|H_0)}{p(\mathbf{X}|H_1)} = \begin{cases} \geq \theta & \text{accept} \\ < \theta & \text{reject} \end{cases} \tag{3.2.1}$$

where

$$H_0 : \mathbf{X} \text{ belongs to speaker } s$$

$$H_1 : \mathbf{X} \text{ does not belong to speaker } s .$$

Hence the task of speaker verification is to find the best way to calculate the two probability density functions, $p(\mathbf{X}|H_0)$ and $p(\mathbf{X}|H_1)$, given the two hypotheses. In GMM, speaker-dependent models are estimated from training speech, thus H_0 can be easily found. However, the other hypothesis where $\mathbf{X}$ is not from s is less well defined, which leads to the solution of using a UBM.

In this approach, a general background GMM is trained through a great amount of speech data from a large number of speakers. And this background model is used to estimate H_1, in which case the likelihood computation function can be written as:

$$\Lambda(\mathbf{X}, M_s) = \frac{p(\mathbf{X}|\mathbf{GMM_s})}{p(\mathbf{X}|\mathbf{UBM})} \tag{3.2.2}$$

and the log-likelihood is:

$$\text{LLK}(\mathbf{X}, \mathbf{GMM_s}) = \log p(\mathbf{X}|\mathbf{GMM_s}) - \log p(\mathbf{X}|\mathbf{UBM}) \tag{3.2.3}.$$

Besides the use of impostor information, UBM also provides a way to derive and update speaker models through Maximum-A-Posteriori (MAP) adaptation. The parameter set

$\theta = \{w_c, \mu_c, \mathbf{\Sigma}_c, c = 1, \ldots, C\}$ for a speaker model remains the same as for a GMM, while the way to estimate it changes. Given a trained C -mixture UBM and a training vector $\mathbf{X} = (x_1, x_2, \ldots, x_F)$, for the c th mixture of UBM, the probabilistic alignment is computed as:

$$\Pr(c \mid x_i) = \frac{w_c p_c(x_i)}{\sum_{k=1}^{C} w_k p_k(x_i)} \tag{3.2.4}$$

Then the zero-, first-, and second-order Baum-Welch statistics are calculated, which are required for computing the weight, mean and variance:

$$N_c(x) = \sum_{i=1}^{F} \Pr(c \mid x_i) \tag{3.2.5}$$

$$F_c(x) = \sum_{i=1}^{F} \Pr(c \mid x_i) x_i \tag{3.2.6}$$

$$S_c(x^2) = \sum_{i=1}^{F} \Pr(c \mid x_i) x_i^2 \tag{3.2.7}.$$

The posterior mean and covariance matrix of the feature components x given the entire observation $\mathbf{X}$ can be found as:

$$E_c(x \mid \mathbf{X}) = \frac{F_c(x)}{N_c} \tag{3.2.8}$$

$$E_c(x^2 \mid \mathbf{X}) = \frac{S_c(x)}{N_c(x)} \tag{3.2.9}.$$

Speaker model parameters are obtained by solving the following equations:

$$\tilde{w}_c = [\frac{\alpha_c n_c}{F} + (1 - \alpha_c) w_c] \cdot \gamma \tag{3.2.10}$$

$$\tilde{\mu}_c = \alpha_c E_c(x \mid \mathbf{X}) + (1-\alpha_c)\mu_c \tag{3.2.11}$$

$$\tilde{\boldsymbol{\Sigma}}_c = \alpha_c E_c(x^2 \mid \mathbf{X}) + (1-\alpha_c)(\boldsymbol{\Sigma}_c + \mu_c^2) - \tilde{\mu}_c^2 \tag{3.2.12}$$

where γ is a scaling factor that ensures all mixture weights sum to 1 and α_c is called the adaptation coefficient which is defined as:

$$\alpha_c = \frac{n_c}{n_c + r} \tag{3.2.13}$$

where r is a fixed relevance factor. This adaptation coefficient helps to determine the importance of each mixture. n_c is defined as the probabilistic count of $\mathbf{X}$ generated from mixture c, therefore α_c would be close to zero when n_c is low and close to one when n_c is high. In other words, if $\mathbf{X}$ is not likely to be drawn from one mixture, the importance of the updated parameters would be minimized. And if there is a high probability that $\mathbf{X}$ comes from one mixture, this approach would emphasize the new parameters as they can describe the features better. This strategy is especially beneficial to limited training data. The GMM-UBM modeling in speaker verification can be summarized in Table 3-2.

Table 3-2: Modeling process of GMM-UBM

Development data training:
1. Given speech from a large number of speakers, train a UBM.
Model training:
1. Given a set of training features of speaker $S: \mathbf{X} = (x_1, x_2, \ldots, x_F)$, initialize $p(x)$ from random θ using (3.1.1) & (3.1.2).
2. Iterate and update θ using (3.2.10), (3.2.11) and (3.2.12).
Testing:
1. Given a speaker model $\mathbf{GMM_s}$, a background model $\mathbf{UBM}$ and an observation $\mathbf{X}$, calculate the likelihood according to (3.2.3).
2. Make a decision based on the likelihood.

3.3 GMM supervector based SVM

3.3.1 GMM supervector

After the success of GMM-UBM systems, latent factor analysis was used for addressing the speaker and channel variability in speaker verification (Kenny & Dumouchel, 2004b). These works use latent factors to model the adapted-GMM means, then estimate the variabilities from GMM means. GMM supervector was developed from this approach which concatenate the means of each mixture component from an adapted-GMM.

Given a C-mixture UBM and a training vector $\mathbf{X} = (x_1, x_2, \ldots, x_F)$ from speaker s, a C-mixture speaker GMM is adapted. The supervector for $\mathbf{X}$ is obtained as:

$$M(s) = [m_1 \quad m_2 \quad \cdots \quad m_C]^{\mathrm{T}} \tag{3.3.1}$$

where $m_c, \quad 1 < c \leq C$ is the mean of the c^{th} mixture. In this approach, each utterance is simply represented by a single vector of a fixed dimension C, which is perfectly suitable for SVM process.

3.3.2 Support Vector Machines

SVM is a widely used binary classification method proposed by Vapnik and Lerner (Cortes & Vapnik, 1995). It has been extended to non-linear frameworks with the introduction of kernel functions (Piciarelli, Micheloni, & Foresti, 2008). Consider a dataset consisting of N feature vectors $\mathbf{X} = (x_1, x_2, \ldots, x_N)$ and these features belong to either of two class, w_1 and w_2. First assume that the data from the two classes are linearly separable, therefore the task of the SVM is to build a hyperplane that can correctly separate all data:

$$g(x) = \mathbf{w}^{\mathrm{T}} x + w_0 \tag{3.3.2}$$

where $\mathbf{w}$ is a normal vector and w_0 is a bias term. The classification task is illustrated in Figure 3-2 (a), where two types of training data are presented as well as two possible hyperplanes.

It is reasonable to say that hyperplane A is better than B because it leaves more room for both sides. In other words, hyperplane A allows training data to move more freely then hyperplane B, while not classifying data into the wrong class.

The term margin is defined as the room that the hyperplane leaves for both classes. In SVM, the optimum solution is to find a hyperplane that has the same distance from the respective nearest points in w_1 and w_2, while maximizing the margin. For further analysis, $\mathbf{w}$ and w_0 can be scaled so that the value of $g(x)$ equals 1 for w_1 and -1 for w_2 at the nearest points in w_1, w_2 (circled in Figure 3-2 (b)) (Theodoridis & Koutroumbas, 2009).

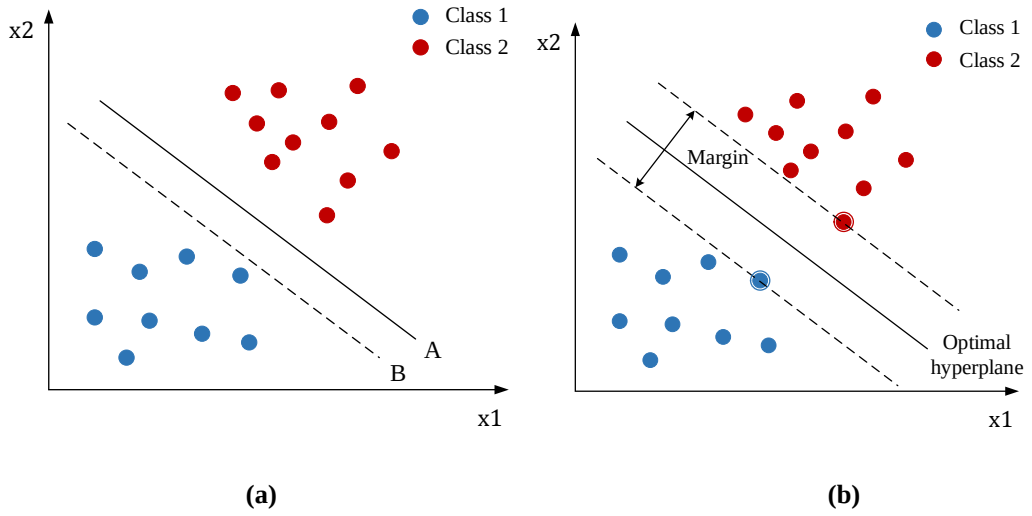

Figure 3-2: An example of a linearly separable two-class problem. (a) with two possible hyperplanes. (b) with optimum hyperplane.

When the data from two classes are not linear-separable, SVM maps the original data into a higher dimensional space $\Re$ through kernel functions. A kernel function is defined as:

$$\kappa(x_i, x_j) = \left\langle \phi(x_i), \phi(x_j) \right\rangle \tag{3.3.3}$$

where $\langle\ \rangle$ stands for dot product and $\phi(x_i)$ is a mapping function. Some common kernels used in SVM are linear, Gaussian and Radial Basis Function (RBF) functions. Regarding GMM supervectors, the task for SVM is to separate two utterances utt_a and utt_b which leads to the problem of finding a kernel $\kappa(utt_a, utt_b)$.

3.3.3 GMM supervector SVM linear kernel

In the case of a simple linear kernel, the function $\phi(x_i)$ is just mapping the utterance of the corresponding supervector:

$$\kappa(utt_a, utt_b) = \left(M(a)\right)^{\mathrm{T}} M(b) \tag{3.3.4}$$

Two new supervector-based kernels were introduced: the Super Linear kernel and Super L^2 kernel, namely:

$$\kappa_L(utt_a, utt_b) = \sum_{j=1}^{K} (\sqrt{w_j}\boldsymbol{\Sigma}_j^{-1/2} m_j^a)^{\mathrm{T}} (\sqrt{w_j}\boldsymbol{\Sigma}_j^{-1/2} m_j^b) \tag{3.3.5}$$

$$\kappa_{L^2}(utt_a, utt_b) = \sum_{j=1}^{K} w_j^2 N(m_j^a - m_j^b; 0, 2\boldsymbol{\Sigma}_j) \tag{3.3.6}$$

where w_j, m_j and $\boldsymbol{\Sigma}_j$ stand for the mixture weight, mean and covariance matrix respectively. The GMM-supervector SVM modeling in speaker verification can be summarized in Table 3-3.

Table 3-3: Modeling process of GMM-supervector

Development data training:
1. Given speech from a large number of speakers, train a UBM.
Model Training:
1. Build an adapted-GMM model for each training utterance from speaker s .
2. Compute a supervector for each utterance by concatenating the means of all mixture component.
3. Choose a suitable kernel function, then use all supervectors from speaker s and nearly the same amount of supervectors from imposters to train a speaker-SVM.
Testing:
1. Compute the supervector of a testing utterance.
2. Calculate the decision by equation (3.3.3) with the testing supervector.

3.4 Joint Factor Analysis

Factor Analysis (FA) is a statistical method used for describing the variabilities in observations. In speaker verification this strategy is employed to discover the speaker- and channel-dependent variabilities in speech (Kenny & Dumouchel, 2004a). Reconsidering the MAP adaptation in the GMM-UBM approach, equation (3.2.11) can be decomposed into two components: a speaker-dependent part $\alpha_c E_c(x \mid \mathbf{X})$ and a speaker-independent part $(1-\alpha_c)\mu_c$. Thus, a GMM-supervector can be written as:

$$M(s) = m + \mathbf{D}z(s) \tag{3.4.1}$$

where m is a speaker-independent supervector, $\mathbf{D}$ is a $CF \times CF$ diagonal residual matrix and $z(s)$ is a $CF \times 1$ hidden vector which follows a normal distribution, $N(z \mid 0, \mathbf{I})$. Consider a speech feature vector $\mathbf{X}_h(s)$ from the h^{th} session of speaker s. JFA follows an assumption that the speaker-dependent supervector is a linear combination of four components (Kenny, 2005):

$$M_h(s) = m + \mathbf{V}y(s) + \mathbf{U}x_h(s) + \mathbf{D}z(s) \tag{3.4.2}$$

where two new terms are added:

1. $\mathbf{V}y(s)$ is the speaker-dependent component, where $\mathbf{V}$ is a $CF \times R_S$ matrix called the eigenvoice matrix (R_S is the number of speaker factors), y_s is a hidden vector that follows a normal distribution ($R_S \times 1$) and is called speaker factors.
2. $\mathbf{U}x_h(s)$ is the channel-dependent component, where $\mathbf{U}$ is a $CF \times R_C$ matrix called the eigenchannel matrix (R_C is number of channel factors). $x_h(s)$ is a hidden vector that follows a normal distribution ($R_C \times 1$), also known as the channel factors.

JFA training includes the estimation of parameter set $\theta = \{m, \mathbf{V}, \mathbf{U}, \mathbf{D}, \mathbf{\Sigma}\}$, where $\mathbf{\Sigma}$ stands for a $CF \times CF$ diagonal covariance matrix with diagonal blocks equal to $\mathbf{\Sigma}_c$, $c = 1, ..., C$. It is assumed that for all observations drawn, the mixture component c follows a Gaussian distribution $N(\sim | M_{hc}(s), \mathbf{\Sigma}_c)$, where $M_{hc}(s)$ is the subvector of $M_h(s)$ related to component c. The core idea behind JFA training is to find an optimal θ that best describes the speaker- and channel-variabilities among recordings. This can be done by increasing the likelihood $\Pr(x_i)$ through iterations given any observation x_i. Similar to MAP optimization in the GMM adaptation procedure, several sufficient statistics are required for training. Based on Equations (3.2.5) to (3.2.7):

$$\tilde{F}_c(s) = F_c(s) - N_c(s) m_c \tag{3.4.3}$$

$$\tilde{S}_c(s) = S_c(s) - diag(F_c(s) m_c^* + m_c F_c(s)^* - N_c(s) m_c m_c^*) \tag{3.4.4}$$

where m_c is the mean vector of the c^{th} component from a UBM model. The statistical matrices across all mixture components are defined as:

$$\mathbf{N}(s) = \begin{bmatrix} N_1(s) \cdot I & & \\ & \ddots & \\ & & N_C(s) \cdot I \end{bmatrix}_{CF \times CF} \tag{3.4.5}$$

$$\mathbf{F}(s)=\begin{bmatrix}\tilde{F}_1(s)\\ \vdots \\ \tilde{F}_C(s)\end{bmatrix}_{CF\times 1} \tag{3.4.6}$$

$$\mathbf{S}(s)=\begin{bmatrix}\tilde{S}_1(s) & & \\ & \ddots & \\ & & \tilde{S}_C(s)\end{bmatrix}_{CF\times CF} \tag{3.4.7}.$$

JFA training is completed upon a development dataset (similar as the one for UBM training) by finding an optimum parameter set θ. With the properly trained parameters and the sufficient statistics of a speech, $y(s)$, $x_h(s)$ and $z(s)$ can be obtained according to their posterior means given the observations:

$$\bar{y}(s)=E\left[y(s)\right]=l_V^{-1}(s)*\mathbf{V}^{*}*\mathbf{\Sigma}^{-1}*\mathbf{F}(s) \tag{3.4.8}$$

$$\bar{x}_h(s)=E\left[x_h(s)\right]=l_U^{-1}(s)*\mathbf{U}^{*}*\mathbf{\Sigma}^{-1}*\mathbf{F}(s) \tag{3.4.9}$$

$$\bar{z}(s)=E\left[z(s)\right]=l_D^{-1}(s)*\mathbf{D}*\mathbf{\Sigma}^{-1}*\mathbf{F}(s) \tag{3.4.10}$$

where

$$\begin{aligned} l_V(s)&=\mathbf{I}+\mathbf{V}^{*}*\mathbf{\Sigma}^{-1}*\mathbf{N}(s)*\mathbf{V}\\ l_U(s)&=\mathbf{I}+\mathbf{U}^{*}*\mathbf{\Sigma}^{-1}*\mathbf{N}(s)*\mathbf{U}\\ l_D(s)&=\mathbf{I}+\mathbf{D}^{2}*\mathbf{\Sigma}^{-1}*\mathbf{N}(s)\end{aligned} \tag{3.4.11}.$$

Scores are calculated based on the given parameter set along with the target and test observations:

$$score=(\mathbf{V}y_{target}+\mathbf{D}z_{target})^{*}\cdot\mathbf{\Sigma}^{-1}\cdot(\mathbf{F}_{tst}-\mathbf{N}_{tst}m-\mathbf{N}_{tst}\mathbf{U}x_{tst}) \tag{3.4.12}.$$

The JFA matrices training procedure can be summarized as in Table 3-4. Detailed iteration formulas can be found in (Kenny, 2005).

Table 3-4: Modeling process of JFA

Development data training:
1. Train the eigenvoice matrix $\mathbf{V}$, assuming $\mathbf{U}$ and $\mathbf{D}$ are zero.
2. Train the eigenchannel matrix $\mathbf{U}$ given $\mathbf{V}$, assuming $\mathbf{D}$ is zero.
3. Train the residual matrix $\mathbf{D}$ given $\mathbf{V}$ and $\mathbf{U}$.
Model training:
1. Given a target observation, calculate the sufficient statistics $\mathbf{N}_{target}$ and $\mathbf{F}_{target}$.
2. Estimate $y(s)$, $x_h(s)$ and $z(s)$ from $\mathbf{V}$, $\mathbf{U}$ and $\mathbf{D}$ by Equations (3.4.8) to (3.4.10).
Testing:
1. Given a test utterance, calculate the sufficient statistics $\mathbf{N}_{tst}$ and $\mathbf{F}_{tst}$.
2. Estimate $y(s)$, $x_h(s)$ and $z(s)$ from $\mathbf{V}$, $\mathbf{U}$ and $\mathbf{D}$.
3. Given a target, compute the score by Equation (3.4.12).

3.5 I-vector approaches

3.5.1 I-vector

The heavy computational complexity requirements of JFA made it impractical in real life application. Moreover, it has been proved in (Dehak, 2009) that the channel-dependent component in JFA also contains some speaker information. Thus, a new approach called i-vector was proposed, which defines only one "total variability" space (Dehak et al., 2011). In other words, instead of the two components (speaker- and channel-dependent components) described in JFA, i-vector introduces a new component named total variability that contains both speaker and channel information. In this case, a speaker- and channel-dependent supervector for speaker s is written as:

$$M(s) = m + \mathbf{T}w(s) \tag{3.5.1}$$

where m still stands for the speaker- and channel-independent supervector, $\mathbf{T}$ is a low-rank rectangular matrix called total variability matrix (a.k.a. i-vector extractor) and w is the so-called

i-vector that follows a standard Gaussian distribution $N(0,\mathbf{I})$. The estimation procedure of $\mathbf{T}$ is the same as estimating $\mathbf{v}$ and the corresponding i-vector can be calculated similarly:

$$w = (I + \mathbf{T}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{N}(s)\mathbf{T})^{-1} * \mathbf{T}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\mathbf{F}(s) \quad (3.5.2).$$

Instead of a modelling method, this approach is more like a "feature extraction" process. Each utterance is represented by an i-vector, which can be seen as a new feature of the speech. Several scoring methods based on i-vector have been proposed, among which cosine scoring and PLDA proved to be the most successful two. Table 3-5 summarizes the extraction strategy of i-vector. Scoring methods will be further explained in the next sections.

Table 3-5: Extraction process of i-vector

Development data training:
1. Train the total variability matrix $\mathbf{T}$.
I-vector extraction for enrollment and testing speech:
1. Given a target observation, calculate the sufficient statistics $\mathbf{N}(s)$ and $\mathbf{F}(s)$.
2. Estimate i-vector w from $\mathbf{T}$ according to Equation (3.5.2).

3.5.2 Cosine distance scoring

Since the i-vector extraction process is exactly the same for enrollment data and testing data, the scoring stage can be simply conducted by comparing the similarity between two i-vectors. It has been proven that the magnitude of an i-vector does not contain any speaker or channel information. Therefore the cosine distance scoring technique was proposed in (N. Dehak et al., 2011), which calculates the angle between two i-vectors:

$$score(w_{\text{target}}, w_{\text{test}}) = \frac{\left\langle w_{\text{target}}, w_{\text{test}} \right\rangle}{\left\| w_{\text{target}} \right\| \left\| w_{\text{test}} \right\|} \quad (3.5.3)$$

3.6 Probabilistic linear discriminant analysis

Probabilistic linear discriminant analysis (PLDA) was first proposed in face recognition (Prince & Elder, 2007). It follows the idea that one face image can be decomposed into two parts:

one which depends only on the identity of people, the other one which describes the within-individual variabilities. In 2011, PLDA was found to produce robust performance in combination with i-vector (Kenny, 2010). The theory behind PLDA is similar as JFA, which describes the speech information with a speaker-dependent part and a channel-dependent part. Given an i-vector w_{ij} extracted from the j^{th} speech of speaker i, PLDA assumes that it can be written as:

$$w_{ij} = m + \mathbf{V}z_i + \varepsilon_{ij} \tag{3.6.1}$$

where m is the global mean of all i-vectors which belong to i, $\mathbf{V}$ is a rectangular matrix and z_i stands for the speaker factor of speaker i. ε_{ij} is a residual term that follows a Gaussian distribution $N(0,\Sigma)$. All speech frames from the same speaker i should share the same speaker factors z_i. Assume that there are H_i speech frames from speaker i in total, then the collection of all speech frames from i can be represented as:

$$\hat{w}_i = \hat{m} + \hat{\mathbf{V}}z_i + \hat{\varepsilon}_i \tag{3.6.2}$$

where $\hat{w}_i = \begin{bmatrix} w_{i1}^{\mathrm{T}} & \cdots & w_{iH_i}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, $\hat{m}_i = \begin{bmatrix} m^{\mathrm{T}} & \cdots & m^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, $\hat{\mathbf{V}}_i = \begin{bmatrix} \mathbf{V}^{\mathrm{T}} & \cdots & \mathbf{V}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ and $\hat{\varepsilon}_i = \begin{bmatrix} \varepsilon_{i1}^{\mathrm{T}} & \cdots & \varepsilon_{iH_i}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$.

It can be seen from Equation (3.6.2) that three parameters are needed to complete a PLDA model, which is denoted as a parameter set $\theta = \{m, \mathbf{V}, \mathbf{\Sigma}\}$. The task of PLDA training is to find an optimal θ_{opt} that can best describe the within- and between- individual variabilities, which is normally done through the EM algorithm. The estimation process can be summarized as below:

1. Initialization: Randomly initialize $\theta = \{m, \mathbf{V}, \mathbf{\Sigma}\}$.
2. E-step:

$$\mathbf{L}_i = \mathbf{I} + H_i \mathbf{V}^{\mathrm{T}} \mathbf{\Sigma}^{-1} \mathbf{V} \tag{3.6.3}$$

$$\left\langle z_i \mid \hat{w}_i \right\rangle = \mathbf{L}_i^{-1}\mathbf{V}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\sum_{j=1}^{H_i}(w_{ij}-m) \tag{3.6.4}$$

$$\left\langle z_i z_i^{\mathrm{T}} \mid \hat{w}_i \right\rangle = \mathbf{L}_i^{-1} + \left\langle z_i \mid \hat{w}_i \right\rangle \left\langle z_i \mid \hat{w}_i \right\rangle^{\mathrm{T}} \tag{3.6.5}$$

3. M-step:

$$\tilde{m} = \frac{\sum_{ij} w_{ij}}{\sum_i H_i} \tag{3.6.6}$$

$$\tilde{\mathbf{V}} = \left[\sum_{ij}(w_{ij}-\tilde{m})\left\langle z_i \mid \hat{w}_i \right\rangle^{\mathrm{T}}\right]\left[\sum_{ij}\left\langle z_i z_i^{\mathrm{T}} \mid \hat{w}_i \right\rangle\right]^{-1} \tag{3.6.7}$$

$$\tilde{\boldsymbol{\Sigma}} = \frac{1}{\sum_{i=1}^{N} H_i}\left\{\sum_{i=1}^{N}\sum_{j=1}^{H_i}\left[\left(w_{ij}-\tilde{m}\right)\left(w_{ij}-\tilde{m}\right)^{\mathrm{T}} - \tilde{\mathbf{V}}\left\langle z_i \mid \hat{w}_i \right\rangle \left(w_{ij}-\tilde{m}\right)^{\mathrm{T}}\right]\right\} \tag{3.6.8}.$$

Given a well-trained PLDA parameter set, the log likelihood between two i-vectors can be calculated as:

$$LLK(w_s, w_t) = \frac{1}{2}\left[w_s^T \mathbf{Q} w_s + 2 w_s^T \mathbf{P} w_t + w_t^T \mathbf{Q} w_t \right] + const \tag{3.6.9}$$

$$\mathbf{Q} = \boldsymbol{\Sigma}_{tot}^{-1} - \left(\boldsymbol{\Sigma}_{tot} - \boldsymbol{\Sigma}_{ac}\boldsymbol{\Sigma}_{tot}^{-1}\boldsymbol{\Sigma}_{ac}\right)^{-1} \tag{3.6.10}$$

$$\mathbf{P} = \boldsymbol{\Sigma}_{tot}^{-1}\boldsymbol{\Sigma}_{ac}\left(\boldsymbol{\Sigma}_{tot} - \boldsymbol{\Sigma}_{ac}\boldsymbol{\Sigma}_{tot}^{-1}\boldsymbol{\Sigma}_{ac}\right)^{-1} \tag{3.6.11}$$

where $\boldsymbol{\Sigma}_{tot} = \mathbf{V}\mathbf{V}^{\mathrm{T}} + \boldsymbol{\Sigma}$ and $\boldsymbol{\Sigma}_{ac} = \mathbf{V}\mathbf{V}^{\mathrm{T}}$.

3.7 Mixture of PLDA

3.7.1 SNR-Independent Mixture of PLDA

The idea of PLDA modelling in the previous section follows an assumption that an i-vector can be described by a single Gaussian distribution. However, researchers suggested that one Gaussian may not be sufficient to model an i-vector due to the variabilities contained in it. Thus, Mak et al. proposed an extension of conventional PLDA called SNR-Independent Mixture of PLDA in 2016, which assumes that an i-vector can be seen as a mixture of several Gaussians Mak et al., 2016b). Intuitively, the parameter set becomes:

$$\theta_{SI-mPLDA} = \left\{ \varphi_k, m_k, \mathbf{V}_k, \mathbf{\Sigma}_k \right\}_{k=1}^{K} \tag{3.7.1}$$

Here an additional term φ_k was introduced which indicates the mixture weight of the k th mixture, where K is the total number of mixtures that are used to describe the i-vector. Following the same EM formulation, the likelihood ratio of two i-vectors in this case can be calculated as:

$$LR_{SI-mPLDA}(w_s, w_t) = \frac{\sum_{k_s=1}^{K}\sum_{k_t=1}^{K} \varphi_{k_s}\varphi_{k_t} \mathrm{N}\left(\begin{bmatrix} w_s^{\mathrm{T}} & w_t^{\mathrm{T}} \end{bmatrix} | \begin{bmatrix} w_{k_s}^{\mathrm{T}} & w_{k_t}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}, \widehat{\mathbf{V}}_{k_s k_t} \widehat{\mathbf{V}}_{k_s k_t}^{\mathrm{T}} + \widehat{\mathbf{\Sigma}}_{k_s k_t} \right)}{\left[\sum_{k_s=1}^{K} \varphi_{k_s} \mathrm{N}\left(w_s \mid m_{k_s}, \mathbf{V}_{k_s}\mathbf{V}_{k_s}^{\mathrm{T}} + \Sigma_{k_s} \right) \right] \left[\sum_{k_t=1}^{K} \varphi_{k_t} \mathrm{N}\left(w_t \mid m_{k_t}, \mathbf{V}_{k_t}\mathbf{V}_{k_t}^{\mathrm{T}} + \mathbf{\Sigma}_{k_t} \right) \right]} \tag{3.7.2}$$

where $\widehat{\mathbf{\Sigma}}_{k_s k_t} = \mathrm{diag}\left\{ \mathbf{\Sigma}_{k_s}, \mathbf{\Sigma}_{k_t} \right\}$ and $\widehat{\mathbf{V}}_{k_s k_t} = \begin{bmatrix} \mathbf{V}_{k_s}^{\mathrm{T}} & \mathbf{V}_{k_t}^{\mathrm{T}} \end{bmatrix}$.

3.7.2 SNR-Dependent Mixture of PLDA

In order to further address the noise condition mismatch between training and testing speech, another extension was developed which contains SNR information (Mak et al., 2016b): the SNR-Dependent Mixture of PLDA (SD-mPLDA). A new parameter describing the SNR information was added:

$$\theta_{SD-mPLDA} = \left\{ \lambda_k, w_k \right\}_{k=1}^{K} = \left\{ \pi_k, \mu_k, \sigma_k, m_k, \mathbf{V}_k, \mathbf{\Sigma}_k \right\}_{k=1}^{K} \tag{3.7.3}$$

where $\lambda_k = \{\pi_k, \mu_k, \sigma_k\}$ stands for the prior probability, mean and standard deviation of the SNR for the k th mixture. Moreover, it can be noticed that there is no mixture weight in this case, which is because the clustering of i-vectors in this case is guided by the SNR information. In other words, it is the posterior probabilities of SNR that determines which mixture an i-vector belongs to instead of the posterior probabilities of i-vectors themselves. As in previous methods, the parameters of SD-mPLDA are obtained through the EM algorithm and the likelihood between two i-vectors is computed as:

$$LR_{SD-mPLDA}(w_s, w_t) = \frac{\sum_{k_s=1}^{K}\sum_{k_t=1}^{K}\gamma_{l_s,l_t}\left(y_{k_s}, y_{k_t}\right)\mathrm{N}\left(\left[w_s^{\mathrm{T}} \quad w_t^{\mathrm{T}}\right]^{\mathrm{T}} | \left[m_{k_s}^{\mathrm{T}} \quad m_{k_t}^{\mathrm{T}}\right]^{\mathrm{T}}, \hat{\mathbf{V}}_{k_s k_t}\hat{\mathbf{V}}_{k_s k_t}^{\mathrm{T}} + \hat{\boldsymbol{\Sigma}}_{k_s k_t}\right)}{\left[\sum_{k_s=1}^{K}\gamma_{l_s}\left(y_{k_s}\right)\mathrm{N}\left(w_s | m_{k_s}, \mathbf{V}_{k_s}\mathbf{V}_{k_s}^{\mathrm{T}} + \boldsymbol{\Sigma}_{k_s}\right)\right]\left[\sum_{k_t=1}^{K}\gamma_{l_t}\left(y_{k_t}\right)\mathrm{N}\left(w_t | m_{k_t}, \mathbf{V}_{k_t}\mathbf{V}_{k_t}^{\mathrm{T}} + \boldsymbol{\Sigma}_{k_t}\right)\right]} \tag{3.7.4}$$

$$\gamma_{l_s,l_t}\left(y_{k_s}, y_{k_t}\right) = \frac{\pi_{k_s}\pi_{k_t}\mathrm{N}\left(\left[l_s \quad l_t\right]^{\mathrm{T}} | \left[\mu_{k_s} \quad \mu_{k_t}\right]^{\mathrm{T}}, \mathrm{diag}\left\{\sigma_{k_s}^2, \sigma_{k_t}^2\right\}\right)}{\sum_{k_s=1}^{K}\sum_{k_t=1}^{K}\pi_{k_s}\pi_{k_t}\mathrm{N}\left(\left[l_s \quad l_t\right]^{\mathrm{T}} | \left[\mu_{k_s} \quad \mu_{k_t}\right]^{\mathrm{T}}, \mathrm{diag}\left\{\sigma_{k_s}^2, \sigma_{k_t}^2\right\}\right)} \tag{3.7.5}$$

where y_{ijk} indicates if the i-vector w_{ij} belongs to mixture k, $y_{ijk} = 1$ means yes and 0 means no, and l_{ij} is the SNR of the corresponding utterances.

Chapter 4 Inter-session Compensation

4.1 I-vector Length Normalization

Since i-vector/PLDA approaches all assume that i-vectors follow Gaussian distributions, a transform strategy was proposed to address the non-Gaussian behaviour of i-vectors (Romero & Wilson, 2011). According to the basic PLDA formulation:

$$w_{ij} = m + \mathbf{V}z_i + \varepsilon_{ij} \tag{4.1.1}$$

where z_i includes both speaker- and channel- information, the i-vector w_{ij} can be seen as an affine transformation which follows a multivariate Student's T distribution (Romero & Wilson, 2011). A technique called Radial Gaussianization can transform this distribution into a Gaussian distribution (Lyu & Simoncelli, 2009), and the corresponding process contains two steps: 1. whitening the data using the development data statistics; 2. projecting the whitened data into the unit circle.

4.2 Within Class Covariance Normalization

Within Class Covariance Normalization (WCCN) is a compensation approach first applied in SVM solutions (Hatch et al., 2006). Speaker verification normally uses generalized linear kernel functions in SVM approaches, which can be formulated as:

$$k(x_1, x_2) = x_1^{\mathrm{T}} \mathbf{R} x_2 \tag{4.2.1}$$

where $\mathbf{R}$ is a positive semidefinite matrix. WCCN introduces a within-class covariance matrix so that:

$$\mathbf{W}^{-1} = \mathbf{R} \tag{4.2.2}$$

where $\mathbf{w}$ contains the distinguishing information between individuals. Given a large set of development data, the optimal $\mathbf{w}$ is found under the guidance of speaker labels which can be computed as:

$$\mathbf{W} = \frac{1}{S}\sum_{i=1}^{S}\frac{1}{h_i}\sum_{j=1}^{h_i}\left(w_{ij} - \bar{w}_i\right)\left(w_{ij} - \bar{w}_i\right)^{\mathrm{T}} \tag{4.2.3}$$

where S is the number of speakers and h_i is the utterance number of speaker i. $\bar{w}_i$ stands for the mean i-vector of speaker i. Moreover, a transformation function $\mathbf{B}$ can be defined as $\mathbf{W}^{-1} = \mathbf{B}\mathbf{B}^{\mathrm{T}}$ such that this approach can be applied as a simple linear transform. Given an i-vector, the updated vector is:

$$\hat{w} = \mathbf{B}^{\mathrm{T}} w \tag{4.2.4}.$$

4.3 Linear Discriminant Analysis

LDA is a widely used dimension reduction method which also deals with the variances. It projects the data into new orthogonal axes which maximize the distinctions between classes. These orthogonal axes are found by minimizing the within-speaker variance and maximizing the between-speaker variance at the same time. The within-class variance matrix follows the same formulation as in the WCCN, while the between-class variance matrix is introduced:

$$\mathbf{S}_w = \frac{1}{S}\sum_{i=1}^{S}\frac{1}{h_i}\sum_{j=1}^{h_i}\left(w_{ij} - \bar{w}_i\right)\left(w_{ij} - \bar{w}_i\right)^{\mathrm{T}} \tag{4.3.1}$$

$$\mathbf{S}_b = \sum_{i=1}^{S}\left(w_i - \bar{w}\right)\left(w_i - \bar{w}\right)^{\mathrm{T}} \tag{4.3.2}$$

where $\bar{w}$ equals a null vector in the case of speaker verification, since i-vectors follow a standard Gaussian distribution with zero mean. Similarly, a simple transformation matrix can be defined so that $\hat{w} = \mathbf{A}^{\mathrm{T}} w$, where $\mathbf{A}$ is composed of the eigenvectors v from:

$$\mathbf{S}_b v = \lambda \mathbf{S}_w v \tag{4.3.3}$$

in which λ is the diagonal matrix of the eigenvalues. In i-vector approach, LDA is normally applied before WCCN to reduce the dimension of the i-vectors, therefore reducing the computational complexity during the testing stage.

Chapter 5 Experiments

5.1 Experimental Setup

In order to evaluate the performance of the previously described algorithms, this thesis used the LibriSpeech ASR corpus (Openslr.org, n.d.) to conduct all experiments. LibriSpeech is an open-access corpus of English novel reading speech, and the speech utterances vary from 3 sec. to 16 sec.. 7590 clean utterances from 759 speakers were used as a development dataset for training a UBM model, the i-vector extractor, the PLDA model and so on.

In order to model a small-set system meanwhile reducing the chances that most speakers sound either extremely similar or different, 30 speakers were chosen to be target speakers, from which 10 speech utterances for each speaker were used in the model training stage. There were 10 additional speakers considered as impostors, in other words, strangers from outside the target speakers. There was no overlap between the development dataset and the evaluation set. Ten utterances from each speaker were used for testing. For experiments that didn't consider varying the length of the available speech, both training and testing speech utterances were 11 sec.. The 11 sec. utterances were segmented from speech longer than 11 sec. in order to keep the continuity in the speech.

The evaluation technique used in this thesis is the Equal Error Rate (EER), which indicates that the False Rejection Rate (FRR) equals the False Acceptance Rate (FAR). In this experimental setup, a false acceptance implies a speech which is classified to belong to a wrong speaker, no matter whether the speaker is a member of the target speakers or an impostor. Similarly, a false rejection is when a speech from a speaker is rejected to belong to that same, true speaker. In total, there were 300 target trials and 11700 non-target trials, including 3000 impostor trials.

Clean speech is not sufficient to examine the robustness of each method, thus additional noise is added to the clean speech. In order to model the multi-noise and multi-SNR condition, four types noise from Freesound.org (Freesound.org, n.d.) were added to clean speech: babble, car, office and airplane noise. More specifically, the airplane noise is only included in the testing phase

in order to model an unanticipated noise case. Noises were added to clean speech under ITU P.56 (active speech level meter) through VOICEBOX. In experiments which require to estimate the SNR of speech data, SNRs were computed according to the speech level difference between speech parts and silent parts through a Voice Activity Detection (VAD) in VOICEBOX:

$$\begin{aligned} SNR_{dB} &= P_{speech,dB} - P_{noise,dB} \\ &= L_{speech,dB} - L_{silent,dB} \end{aligned} \tag{5.1.1}$$

where $P_{speech,dB}$ and $P_{noise,dB}$ are the power of speech and noise in dB, while $L_{speech,dB}$ and $L_{speech,dB}$ stand for the active speech level according to ITU P.56.

1. Noisy development data: Clean speech were randomly combined with babble, car or office noise at 5 dB to 20 dB SNR. There is only one noisy development dataset which contains speech in three noise conditions and different SNR situations. Figure 5-1 presents the SNR distribution of the noisy development dataset. Speech length varies from 3 sec. to 16 sec. in this dataset.
2. Noisy enrollment data: The enrollment speech was generated like the development data, and there is also only one noisy enrollment dataset where training speech were at different noise and SNR conditions. The SNR distribution of the enrollment data appears in Figure 5-2.
3. Testing data: Clean testing data were combined with babble, car, office and airplane noise at 0 dB, 6 dB, 10 dB and 20 dB respectively, resulting in 16 testing conditions (subsets) in total. Testing speech with airplane noise is then a previously unseen situation. Since speech signal with 20 dB SNR is rather clean and speech with higher SNR is considered not practical in real life, a maximum SNR of 20 dB was considered in this thesis.

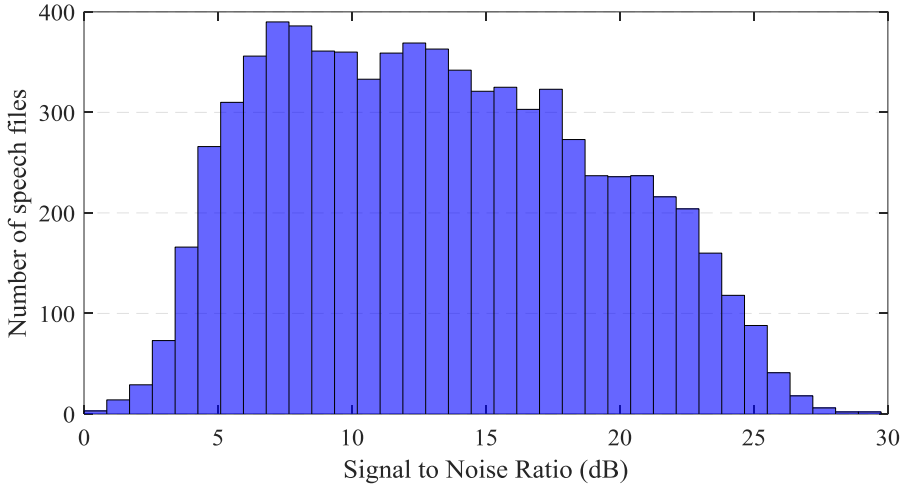

Figure 5-1: SNR distribution of noisy development data.

According to the core idea behind each approach, this thesis divided them into two classes: basic approaches and extension approaches. Basic approaches include GMM, UBM, Supervector/SVM, JFA and conventional PLDA, while extensions contain the i-vector approaches combined with inter-session compensation techniques and mixture of PLDA. Comparisons were first made among basic methods under both clean and mismatch conditions, meanwhile evaluating the performance of four common features: MFCC, LPCC, PLP and RASTA-PLP. Basic approaches with acceptable results will then be compared with extension methods.

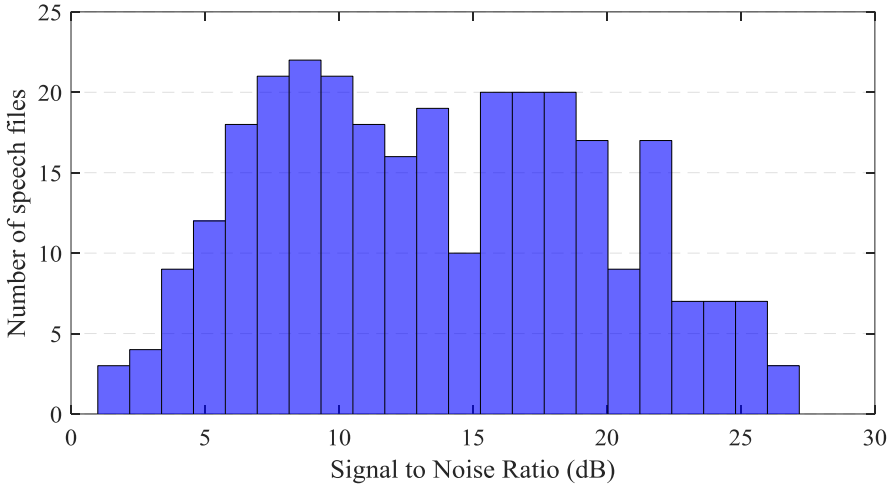

Figure 5-2: SNR distribution of noisy enrollment data.

5.2 Basic Approaches

5.2.1 Clean Condition

This section compares the performance as well as testing time of several algorithms under clean condition (no additive noise). Since there is no channel or environmental effect in this case, no inter-session compensation strategy is applied. The features used for computing i-vectors were 60-dimensional vectors each containing 19 dimensional MFCCs along with their energy plus delta and double delta features (3 times 19+1), extracted with a 25 ms Hamming window, followed by cepstral mean and variance normalization. The experiments are conducted by using an Intel Core i7-4790 CPU (3.60 GHz) with Windows 7 operating system. As MFCC extraction is the same for all approaches, testing time is counted after MFCC extraction of testing speech and ended after completion of 12000 trials. There were five main methods used for testing:

1. GMM: each target speaker is modeled by a 1024 mixture of diagonal GMMs (i.e. GMMs with covariance matrices forced to be diagonal).
2. GMM-UBM: a 1024 mixture of diagonal UBMs is trained from the clean development dataset, then each target speaker is modeled by a 1024 mixture of diagonal adapted-GMM.
3. GMM supervector/SVM: supervectors are extracted based on the GMM/UBM setup. In this case each speech utterance from a speaker was treated independently, thus each target speaker

has 10 supervectors. Additionally, for each target speaker model training, a sub-dataset that contains 400 speakers randomly selected from the development dataset is built. And 400 supervectors are extracted from this subset to form the impostor data in the SVM training phase. In order to balance the data from the target speaker (positive) and the impostors (negative) part in SVM training, supervectors of target speakers are duplicated to reach the same amount of impostor supervectors. Linear kernel and supervector linear kernel are both used.

4. JFA: The JFA model is trained from the clean development data containing 300 speaker factors and 100 channel factors, which follows the setup in (Kenny, Boulianne, Ouellet, & Dumouchel, 2007).
5. I-vector: A clean i-vector extractor with 400 total variability was trained, which lead to 400 dimensional i-vectors. Each speech was represented by one i-vector, resulting in 10 i-vectors for each speaker. Cosine distance scoring and PLDA scoring were both tested. A testing speech was compared against the i-vector averaged across the 10 i-vectors.

Table 5-1 gives the EERs under clean condition. All approaches worked well except the basic GMM, with an EER of less than 1%, especially for the UBM, supervector/SVM with linear kernel and two i-vector methods, which achieved EERs of less than 0.1%. Although the GMM performed the worst, a 2.38% EER may still be acceptable in some scenarios. Regarding the testing time, the UBM appeared to be the most time-expensive method (nearly 10 minutes) while the JFA was the fastest. Both i-vector methods can finish in 2 minutes while reaching a 0.06% EER. Thus it can be concluded that all methods reached a reasonable EER, among which Supervector/SVM with linear kernel outperformed the other methods in clean condition, while GMM was the least robust. UBM also worked well but it required the highest computational complexity (execution time).

Table 5-1: EERs (%) and testing time (sec.) of seven approaches under clean condition

Methods	EER (%)	Testing time (sec.)
GMM	2.38	298.65
UBM	**0.05**	596.26
Supervector/SVM (linear)	**0.03**	174.77
Supervector/SVM (supervector linear)	0.29	178.43
JFA	0.33	**80.02**
i-vector/Cosine	**0.06**	116.81
i-vector/PLDA	**0.06**	113.37

5.2.2 Mismatch Conditions

In order to evaluate the mismatch and noisy cases in real life, noisy enrollment speech were used for training speaker models while first testing speech with three anticipated (previously seen) noise. Moreover, experiments in this section used noisy development data for training UBM, PLDA and i-vector extractor (as opposed to clean data). Figure 5-3 to Figure 5-5 provide the EERs (%) of the seven approaches against 4 SNR conditions under babble, car and office noise.

It is notable that the GMM method is significantly less robust than the other approaches in all SNR and all noise conditions. All the other methods reached an EER of less than 2.50% when the speech was relatively clean (20 and 10 dB). While the performance of most approaches degraded quickly when the SNR dropped from 10 dB to 0 dB, the i-vector/PLDA method kept a slow decreasing trend in its performance. The UBM shared a similar performance to the i-vector/Cosine method at 20 dB and 10 dB, however the EERs of the UBM increased a lot when the SNR was 10 dB or less. Table 5-2 gives the verification results averaged across babble, car and office noise. The i-vector/PLDA approach produced the best results with an averaged EER of 1.45% at 0 dB SNR, while the others were 4 to 20 times worse than it. Although the JFA had a similar performance to the Supervector/SVM approach at 20 dB to 6 dB in most cases, it kept a

slower degradation of performance than the Supervector/SVM when the SNR dropped to 0 dB. Thus it is reasonable to say that JFA is more robust under low SNR conditions while the Supervector/SVM is not a very good solution for noisy cases.

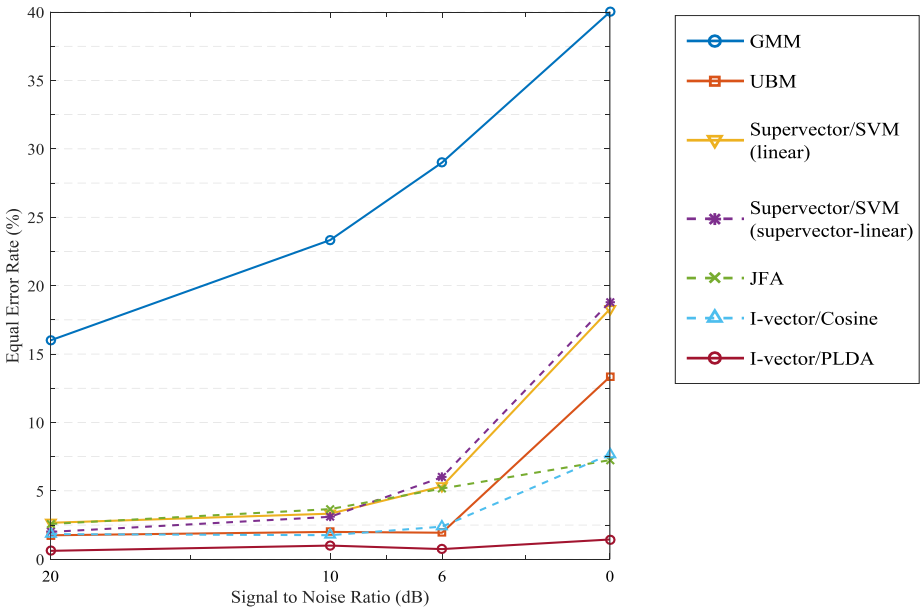

Figure 5-3: EERs (%) of the seven approaches against 4 SNRs under babble noise

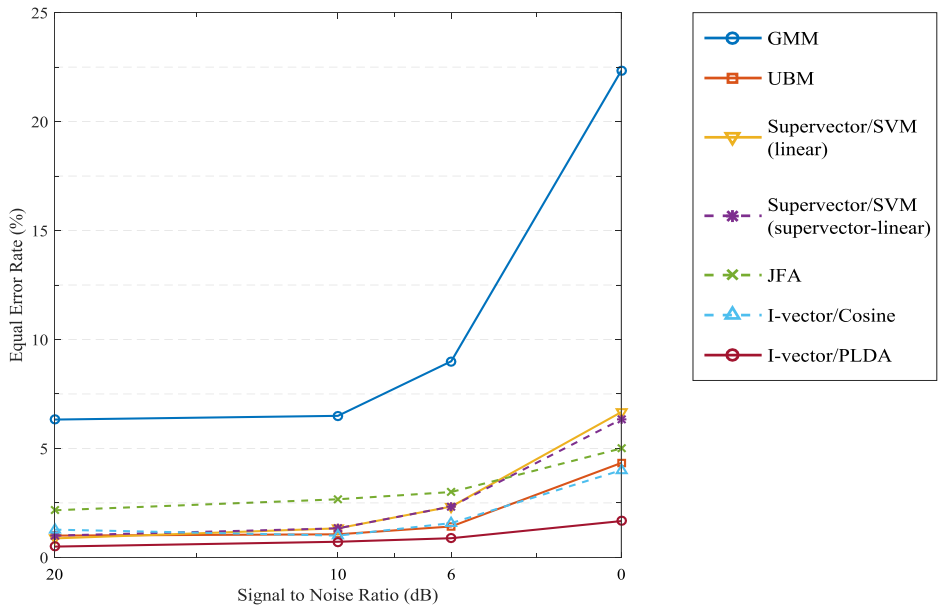

Figure 5-4: EERs (%) of the seven approaches against 4 SNRs under car noise

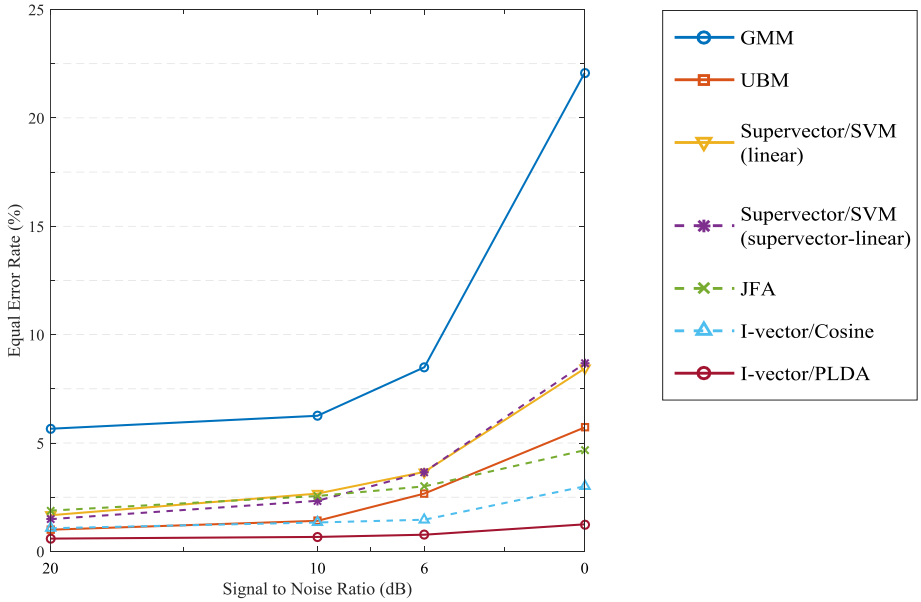

Figure 5-5: EERs (%) of the seven approaches against 4 SNRs under office noise

Table 5-2: EERs (%) of seven approaches averaged across babble, car and office noise

Algorithm	EER (%)			
	20 dB	**10 dB**	**6 dB**	**0 dB**
GMM	11.33	12.03	15.50	28.14
UBM	1.25	1.49	2.01	7.79
Supervector/SVM (linear)	1.74	2.33	3.78	11.13
Supervector/SVM (supervector linear)	1.50	2.26	4.00	11.27
JFA	2.21	2.96	3.72	5.64
i-vector/Cosine	1.39	1.36	1.81	4.89
i-vector/PLDA	**0.57**	**0.79**	**0.80**	**1.45**

In addition, it can be noticed that some methods, such as UBM and i-vector/PLDA with babble noise, worked slightly better under 10 dB SNR than under 20 dB SNR, which is a result due to the limited amount of speech data used in the experiments (10 enrollment speech utterances for each speaker), such that the difference in performance is of the same range as the accuracy of the result. All approaches tend to work better when training and testing speech "match" each other, which means they have the same noise type or are at similar SNR.

5.2.3 Comparison Among Features

In order to make comparable experiments, LPCC, PLP and RASTAPLP features were extracted similarly to the MFCC features. 19 coefficients were extracted from a 25 ms Hamming window with 15 ms overlaps. The features were then appended with their energy plus delta and double delta features, resulting in 60-dim feature vectors, followed by cepstral mean and variance normalization. UBM, JFA and two i-vector approaches were included for this comparison, since they had an acceptable performance in the previous section (EERs less than 8% at 0 dB SNR condition).

With respect to different speech features, Table 5-3 gives the extraction time of each feature. The extraction times were measured via extracting 60-dim coefficients from 400 speech

files. It can be seen that all types of features took similar time, among which extracting the PLP was the fastest. The times spent for extracting features were very short compared to the classification.

Table 5-4 to Table 5-7 show the averaged-EERs of four different features under the three anticipated (previously seen) noise scenarios. MFCC features outperformed the other three features under 20 dB and 0 dB situations with the UBM method, meanwhile the LPCC features achieved the best performance with 10 dB and 6 dB testing SNRs. The PLP features appeared to be less robust than MFCC and LPCC with UBM, and RASTA filtering did not bring any improvement to it. This was predictable since RASTA is supposed to be helpful with additional convolutional noises such as reverberation and microphone response, which are not found in our experiments. With the JFA method, the LPCC features showed an improvement of 0.06% EER at 10 dB, 0.4% to 0.5% EER at 6 dB and 0 dB compared to the MFCC features. The PLP features also gave better results than RASTAPLP, except at 20 dB SNR with the JFA method. For both UBM and JFA methods, MFCC and LPCC features were more robust than PLP and RASTAPLP features in all SNR conditions.

MFCC, LPCC and PLP features produced an imbalanced performance with the i-vector/Cosine approach, while the RASTAPLP features did not show any advantage compared to the other features through all situations. The PLP features produced the best results at high SNR cases (20 dB and 10 dB), meanwhile the LPCC and MFCC features outperform the other methods at 6 dB and 0 dB SNR, respectively. Since the i-vector/PLDA is the most robust approach among all, more focus should be put on this approach. The MFCC features worked the best in most cases except at 10 dB SNR, where the PLP had an improvement of 0.14% EER compared to MFCC. However, LPCC was found to be less robust than MFCC in this case, while RASTA filtering remained the worst. Hence, for the next experiments we will use the MFCC features to evaluate the extension approaches.

Table 5-3: Extraction times (sec.) of four types of feature

Features	Extraction time (sec.)
MFCC	9.85
LPCC	8.00
PLP	**7.49**
RASTAPLP	8.16

Table 5-4: EERs (%) of four features averaged across babble, car and office noise using UBM

Features	EER (%)			
	20 dB	10 dB	6 dB	0 dB
MFCC	**1.25**	1.49	2.01	**7.79**
LPCC	**1.30**	**1.12**	**1.78**	8.47
PLP	1.39	1.84	2.54	8.04
RASTAPLP	1.68	2.17	2.83	9.12

Table 5-5: EERs (%) of four features averaged across babble, car and office noise using JFA

Features	EER (%)			
	20 dB	10 dB	6 dB	0 dB
MFCC	**2.21**	**2.96**	3.72	5.64
LPCC	**2.32**	**2.90**	**3.19**	**5.24**
PLP	3.23	3.73	4.23	5.67
RASTAPLP	3.16	3.87	4.31	5.95

Table 5-6: EERs (%) of four features averaged across babble, car and office noise using i-vector/Cosine

Features	EER (%)			
	20 dB	10 dB	6 dB	0 dB
MFCC	**1.39**	**1.36**	**1.81**	**4.89**
LPCC	**1.39**	1.51	**1.79**	5.77
PLP	**1.27**	**1.35**	2.31	5.44
RASTAPLP	2.38	2.70	3.44	7.25

Table 5-7: EERs (%) of four features averaged across babble, car and office noise using i-vector/PLDA

Features	EER (%)			
	20 dB	10 dB	6 dB	0 dB
MFCC	**0.57**	0.79	**0.80**	**1.45**
LPCC	**0.62**	0.83	**0.91**	1.78
PLP	**0.61**	**0.65**	**0.89**	2.35
RASTAPLP	1.03	1.26	1.32	3.21

5.3 Extension approaches

5.3.1 Anticipated Noises

Based on the previous results, the MFCC feature as well as the UBM, JFA and i-vector approaches were included in this section to compare with four more extensions:

1. I-vector approaches with compensation techniques: compensation methods following the scheme in (Mak et al., 2016a), where WCCN and i-vector length normalization were applied to 400-dim i-vectors, followed by LDA and WCCN reducing the i-vectors dimension to 150. The processed i-vectors were used for Cosine and PLDA scoring.
2. SI-mPLDA and SD-mPLDA: After i-vector extraction, a two mixtures of PLDA approach is employed for scoring, and the same compensation strategies were used since they were reported to improve the performance in (Mak et al., 2016a). Both SI- and SD-mPLDA consist of two Gaussian mixtures, and the number of mixtures is chosen through experiments.

Figure 5-6 to Figure 5-8 give the EER of the 8 approaches against 4 SNRs under babble, car and office noise, i.e., the anticipated, previously seen noises. It is obvious that in all noise cases the i-vector/PLDA and mPLDA approaches worked better than the others. Conventional PLDA outperformed the SI-mPLDA method under babble noise when the SNR was low, while an opposite trend can be observed under office noise, where the SI-mPLDA achieved about 0.3% EER improvement over the conventional PLDA. However, the SD-mPLDA method did not give

any advancement compared to the single-PLDA method, in all cases. As mentioned in the previous section, due to the limited amount of speech data for different SNRs of training speech, the EERs obtained at 6 dB are slightly smaller than 10 dB SNR in some cases.

Table 5-8 provides the EERs (%) averaged through the three anticipated noises for all eight approaches, as well as the corresponding testing time. Inter-session compensation techniques improved the performance of the i-vector/Cosine by 0.36% and 1.83% EER at 20 dB and 0 dB respectively, while using no compensation strategy was more robust at 10 dB and 6 dB SNRs (with improvements of 0.28% and 0.15% EERs). The i-vector/PLDA and i-vector/mPLDA approaches outperformed the others through all SNR conditions, especially at 0 dB. Among these methods, the i-vector/PLDA with compensation strategy achieved the best performance at 20 dB, 10 dB and 6 dB. However, the i-vector/PLDA with no compensation has a slightly better performance by 0.27% EER at 0 dB. Both the SI- and SD-mPLDA methods did not show any improvement compared with conventional PLDA in all conditions.

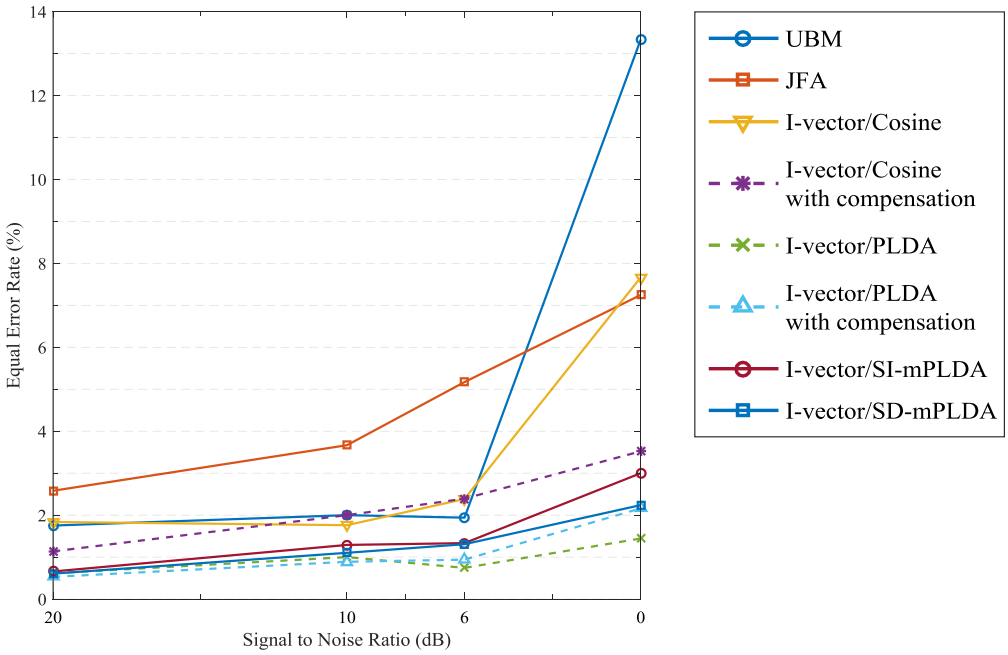

Figure 5-6: EERs (%) of the eight approaches against 4 SNRs under babble noise

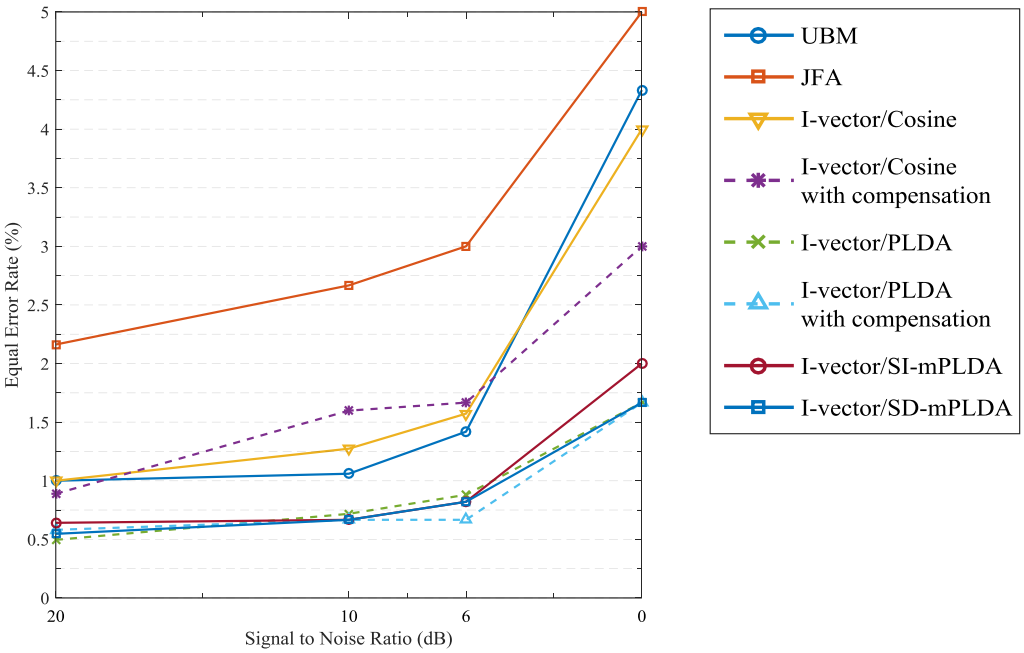

Figure 5-7: EERs (%) of the eight approaches against 4 SNRs under car noise

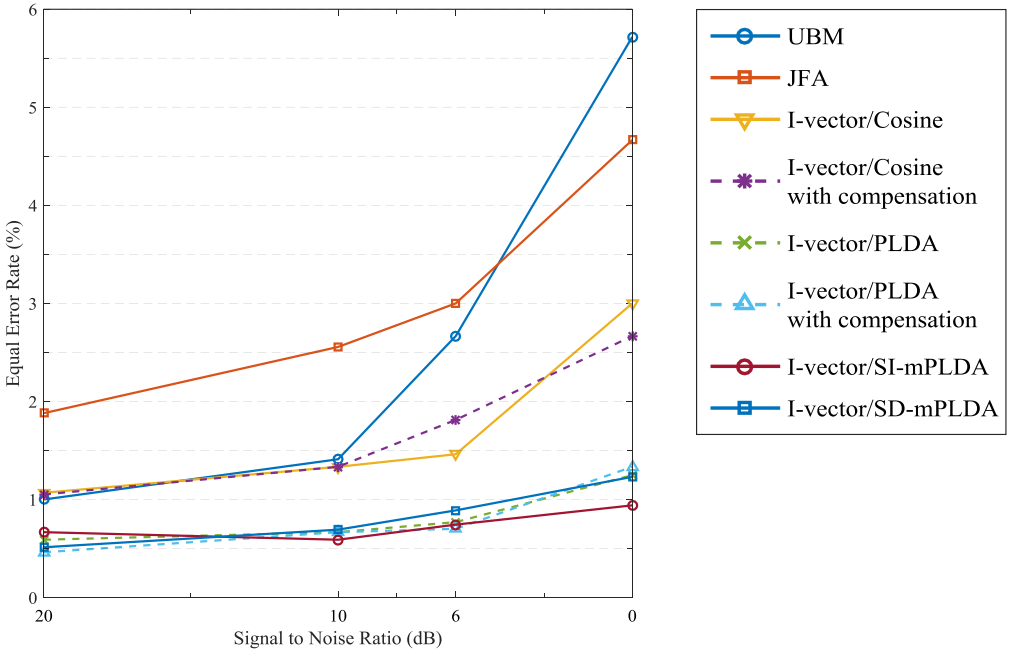

Figure 5-8: EERs (%) of the eight approaches against 4 SNRs under office noise

Table 5-8: EERs (%) of eight approaches averaged across babble, car and office noise

Algorithm	Testing time (s)	EER (%)			
		20 dB	10 dB	6 dB	0 dB
UBM	596.26	1.25	1.49	2.01	7.79
JFA	**80.02**	2.21	2.96	3.72	5.64
i-vector/cosine	116.81	1.39	1.36	1.81	4.89
i-vector/cosine (with compensation)	127.63	1.03	1.64	1.96	3.06
i-vector/PLDA	113.37	**0.57**	**0.79**	**0.80**	**1.45**
i-vector/PLDA (with compensation)	124.19	**0.52**	**0.74**	**0.77**	**1.72**
i-vector/ SI-mPLDA	124.93	**0.66**	**0.85**	**0.97**	**1.98**
i-vector/ SD-mPLDA	518.23	**0.56**	**0.82**	**1.01**	**1.71**

Regarding the time complexity, the compensation strategies only cost about 10 sec. for 12000 trials, thus it is affordable considering the improvements in accuracy that it brought (0.5% EER at 20 dB and 10 dB for i-vector/PLDA). The JFA appears to be the most efficient method but with around 4 times worse EERs compared with i-vector approaches. The SI-mPLDA method shared a similar processing time with conventional PLDA, while the SD-mPLDA method is much more computationally expensive, since the SNR of each testing speech needs to be calculated before scoring. Thus it can be concluded that even though the SD-mPLDA method achieved similar results to the SI-mPLDA (slightly better at 20 dB and 0 dB), it is not a good choice in practice because of its complexity.

5.3.2 Unanticipated Noise

This section analyzes the system performance when utterances with a "previously unseen" noise are processed. Table 5-9 shows the EERs (%) obtained with the eight algorithms when testing utterances produced with unanticipated airplane noise. The performance of all methods degraded compared to the ones with previously known noises, except for the UBM and i-vector/Cosine approaches at 0 dB. JFA gave the worst performance with 8% EER at 0 dB condition, while the i-vector/PLDA with compensation and the i-vector/SD-mPLDA achieved EERs of less than 5%

when testing with 0 dB speech. The I-vector/Cosine approach worked surprisingly well in almost all SNR conditions, with more than 1% EER improvement at low SNRs compared to the second most robust method (i-vector/SD-mPLDA). Compensation techniques were not helpful for Cosine scoring in all cases, however there is 1.5% to 2.5% of improvement in the i-vector/PLDA approach after applying inter-session compensation through all SNRs. Unlike the case with anticipated noises, the SD-mPLDA approach outperformed the SI-mPLDA by 0.5% to 1% in EER for all SNR cases. Hence, it can be summarized that i-vector/Cosine has been the most robust approach when an "unseen" noise was included, and the i-vector/PLDA with compensation and the i-vector/SD-mPLDA methods were also acceptable.

Table 5-9: EERs (%) of eight approaches under unanticipated "airplane" noise

Algorithm	EER (%)			
	20 dB	**10 dB**	**6 dB**	**0 dB**
UBM	2.74	3.00	3.52	6.99
JFA	3.76	5.46	6.67	8.00
i-vector/Cosine	**1.67**	**1.32**	**2.00**	**2.62**
i-vector/Cosine (with compensation)	2.33	3.56	4.33	5.86
i-vector/PLDA	3.62	5.20	5.71	6.29
i-vector/PLDA (with compensation)	2.01	3.26	3.33	4.19
i-vector/SI-mPLDA	3.00	3.47	4.21	5.24
i-vector/SD-mPLDA	1.96	2.90	3.20	4.00

5.4 Score Fusion

Linear score fusion was found to be an efficient way to improve the performance of PLDA approaches (Pang & Mak, 2015), therefore this section examines the robustness of several fusion systems. Given the scores obtained through two methods, the fusion scores are calculated by a linear combination:

$$s_{fusion} = \alpha s_1 + (1-\alpha) s_2,\ 0<\alpha<1 \tag{5.4.1}$$

where s_1 and s_2 stand for the scores from two algorithms, and α is the fusion weight. Since the scores resulting from different approaches lie in different ranges, it is necessary to normalize them before fusion. Zero normalization (a.k.a. Z-norm, Auckenthaler, Carey, & Lloyd-Thomas, 2000) is a commonly used technique in speaker verification. For a test sequence $\mathbf{X}$ and a speaker model $\mathbf{m}$, the normalized score is calculated by:

$$s_{norm}(\mathbf{m},\mathbf{X}) = \frac{\mathrm{LLK}(\mathbf{m},\mathbf{X}) - \mu}{\sigma} \tag{5.4.2}$$

where μ and σ are the mean and standard deviation of the impostor scores against model $\mathbf{m}$. The calculation of μ and σ only requires impostor speech and speaker models, therefore it can be completed offline.

According to the results from previous sections, i-vector/PLDA with compensation was the most robust method to SNR mismatch with previously seen noise. The SI- and SD-mPLDA approaches also had acceptable performances in this case, with less than 1.0% EERs for most SNR situations and EERs within 2% at 0 dB SNR testing condition. Thus the following fusion systems will be evaluated for three anticipated noises scenarios, and the fusion weights will be chosen through the experiments.

1. S1: fusion of i-vector/PLDA with compensation and i-vector/SI-mPLDA, the fusion scores are obtained by Equation (5.4.3) with $\alpha = 0.6$:

$$s_{fusion} = \alpha s_{SI(SD)-mPLDA} + (1-\alpha) s_{PLDA} \tag{5.4.3}$$

2. S2: fusion of i-vector/PLDA with compensation and i-vector/SD-mPLDA, the fusion scores are also from Equation (5.4.3), but with $\alpha = 0.4$.
3. S3: fusion of i-vector/PLDA with compensation, i-vector/SI-mPLDA and i-vector/SD-mPLDA, the fusion scores are according to Equation (5.4.4) with $\alpha_1 = 0.6$, $\alpha_2 = 0.2$:

$$s_{fusion} = \alpha_1 s_{SI-mPLDA} + \alpha_2 s_{SD-mPLDA} + (1-\alpha_1-\alpha_2) s_{PLDA} \tag{5.4.4}.$$

As for the case with unanticipated (not previously seen) airplane noise, the i-vector/Cosine approach was fused with the i-vector/PLDA with compensation (S4) and the i-vector/SD-mPLDA (S5). The fusion scores are computed through Equation (5.4.5) with $\alpha = 0.4$:

$$s_{fusion} = \alpha s_{Cosine} + (1-\alpha) s_{PLDA(SD-mPLDA)} \tag{5.4.5}$$

Table 5-10: EERs (%) of five fusion systems with seen and unseen noises

	Systems	EER (%)			
		20 dB	**10 dB**	**6 dB**	**0 dB**
Averaged across seen noises	S1	0.45	0.73	0.87	1.46
	S2	0.58	0.83	0.94	1.64
	S3	0.45	0.74	0.89	1.44
Unseen noise	S4	1.03	1.25	1.42	2.15
	S5	0.97	1.33	1.33	2.03

The EERs of the five fusion systems are presented in Table 5-10. Compared with the results from the three independent approaches given by Table 5-8, overall the fusion systems showed some small improvements in almost all SNR conditions. In comparison to the results of i-vector/PLDA with compensation, S1 had improvements of 0.07%, 0.01% and 0.26% in EER at 20 dB, 10 dB and 0 dB. S3 also achieved better results at 20 dB and 0 dB (with 0.07% and 0.01% EER improvement, respectively), while fusing with S2 did not bring any benefit. However, the i-vector/PLDA with compensation method is still the most robust approach when testing with 6 dB SNR (with more than 0.1% EER better than fusion systems). With respect to the performance of each fusion system, S1 outperformed the other two in the 10 dB and 6 dB SNR cases. Conversely, S3 was slightly better than S1 when testing with 0 dB SNR speech and they had the same performance in the 20 dB SNR case. It can be concluded that linear fusion is not helpful since the performance gain is very marginal, moreover it will double the computation time from running two algorithms.

When testing with unanticipated airplane noise, a great improvement can be observed compared to the results in Table 5-9. S5 outperformed S4 in most cases except for the 10 dB SNR

case, with 0.1% to 0.3% improvements in EER. This was predictable since the SD-mPLDA was found to be more robust than conventional PLDA with airplane noises. However, SD-mPLDA requires the computation of speech SNR before classifying, thus it is not as practical as conventional PLDA in real life. Therefore, it can be summarized that S4 fusion is an effective way to improve the performance when dealing with previously unseen noise in practice, while S5 can be considered as the best solution when there is no constraint on system complexity.

5.5 Analysis of Testing Length

Since it may be unpleasant for speakers to speak a long utterance to get a verification decision, it is useful to test what is the shortest length of testing speech that can give an acceptable result. Since it is not necessary to consider the performance of ineffective methods, this section only focuses on the most robust methods from the previous sections. The i-vector/PLDA with compensation method and the fusion system S1 were tested with anticipated noise scenarios, while the S4 and S5 fusion methods were evaluated for the previously unseen noise scenario. This section investigates the performance of the four systems when the testing length decreases from 11 sec. to 2 sec., under different SNR conditions.

5.5.1 Anticipated Noise

Despite the robustness of each method, focus is put on the performance impact of the testing speech length. Table 5-11 gives the averaged-EERs of the i-vector/PLDA (with compensation) method. When the SNR is higher than 0 dB, cutting the testing speech length from 8s to 5s produces an EER which is 1.5 to 2 times worse than the EER at 8s. The results are further degraded by a factor around 2.5 times when the testing speech length was cut from 5s to 2s, and all EERs with 2 sec. speech are then greater than 3%. A similar behaviour was observed in the 0 dB SNR case, only the overall performance in this case was twice worse than for higher SNR conditions. The results from the fusion system S1 are presented in Table 5-12, which shows a similar performance trend as Table 5-11. But although S1 previously outperformed the i-vector/PLDA (with compensation) method at 20 dB, 10 dB and 0 dB cases with 11s speech, it was not as robust as the non-fusion system when the testing speech length became shorter.

Table 5-11: EERs (%) of i-vector/PLDA (with compensation) averaged over three seen noises

SNR	EER (%)			
	11s	**8s**	**5s**	**2s**
20 dB	0.52	0.60	1.34	3.29
10 dB	0.74	0.97	1.54	3.90
6 dB	0.77	1.05	1.83	4.56
0 dB	1.72	2.30	3.48	7.84

Table 5-12: EERs (%) fusion system S1 averaged over three seen noises

SNR	EER (%)			
	11s	**8s**	**5s**	**2s**
20 dB	0.45	0.62	1.43	3.91
10 dB	0.73	0.79	1.56	5.33
6 dB	0.87	1.22	1.98	5.88
0 dB	1.46	2.63	3.62	9.33

Regarding the acceptable shortest testing length, using 8 sec. of speech can achieve similar results as 11 sec. of speech in most cases, except for the 0 dB SNR condition where 11-sec. utterances have more than 0.6% improvement in EER over 8-sec. utterances. Although there is a degradation of performance when the testing speech was cut to 5 sec., the performance was still acceptable in most cases: for the i-vector/PLDA with compensation, the EERs were less than 1.9% under 20 dB, 10 dB and 6 dB cases. However, the EERs obtained for 2-sec. conditions were two to three times worse than the ones obtained at the 5-sec. conditions across all SNR conditions and algorithms, therefore it is reasonable to conclude that 2-sec. speech utterances are two short for getting an acceptable verification result. As for 0 dB SNR conditions, 11-sec. utterances are required to get an EER of less than 2% for both systems.

5.5.2 Unanticipated Airplane Noise

The results for the previously unseen “airplane” noise case with the S4 and S5 fusion approaches are given in Table 5-13 and Table 5-14. The performance degraded slowly when the testing speech length was cut from 11sec. to 5 sec., the degradation became larger as the testing

SNR decreased. When the testing speech length was cut to 2 sec., the EERs in all SNR conditions grew rapidly (two times worse than for 5 sec.). Considering the factor of "unseen" noise, it was predictable that the results here would be worse than the ones with previously seen noises. All EERs for the 2 sec. case were higher than 5%, which is not an acceptable error rate. Thus it is reasonable to suggest that speech utterances of at least 5 sec. are necessary at 20 dB to 6 dB SNR, and at least 8 sec. utterances are required at the 0dB SNR condition. But these conditions will not be sufficient to obtain the same EER performance as in Table 5-10 and Table 5-11, since the performance is not as good when testing with a previously unseen noise. When testing with short speech in low SNR cases, the difference between S4 and S5 became smaller.

Table 5-13: EERs (%) of S4 fusion system with airplane noise

SNR	EER (%)			
	11s	**8s**	**5s**	**2s**
20 dB	1.03	2.00	2.33	5.56
10 dB	1.25	2.00	2.21	5.33
6 dB	1.42	2.54	2.67	6.00
0 dB	2.15	2.67	3.74	9.04

Table 5-14: EERs (%) of S5 fusion system with airplane noise

SNR	EER (%)			
	11s	**8s**	**5s**	**2s**
20 dB	0.97	1.86	2.27	5.33
10 dB	1.33	1.85	2.22	5.55
6 dB	1.33	2.33	2.33	6.00
0 dB	2.03	2.76	3.71	9.05

5.6 False Acceptance Rate and False Rejection Rate

5.6.1 Anticipated Noise

In practice there are two other important evaluation metrics for speaker verification systems other than the EER: the false acceptance rate (FAR) and the false rejection rate (FRR). The false acceptance (false positive) is counted as an impostor or noise been recognized as a target

speaker, and a false rejection (false negative) is when a target speaker is rejected by the system. FAR and FRR can be calculated as:

$$\text{FAR} = \frac{\sum \text{False positive}}{\sum \text{Condition negative}} \tag{5.4.6}$$

$$\text{FRR} = \frac{\sum \text{False negative}}{\sum \text{Condition positive}} \tag{5.4.7}$$

where "condition negative" and "condition positive" refer to the actual label of a segment ("ground truth"). In other words, $\sum \text{Condition negative}$ is the total number of impostors and noise segments and $\sum \text{Condition positive}$ is the total number of target speech segments. For a speaker verification system, there is always a trade-off between FAR and FRR. A higher decision threshold will lead to a more "strict" system, which has low FAR and higher FRR. In contrary, a more "loose" system with a lower threshold will have higher FAR and a low FRR. In practice, the decision threshold should be adjusted to meet the system requirements, for example a voice lock system possibly needs a low false acceptance rate to ensure the system security. Based on previous results, the i-vector/PLDA with compensation strategies and S4 fusion are the best approaches for previously seen and unseen noise cases. Therefore, this section will examine the FAR-FRR trade-off of these two methods under three known noises and previously unseen airplane noise conditions, separately. Detection Error Trade-off (DET) curves are a typical way to visualize the relationship between FAR and FRR. Figure 5-9 to Figure 5-11 present the DET curves of the results obtained from i-vector/PLDA with compensation techniques when testing with babble, car and office noises. The black dot lines in the figures indicate the place where the FAR equals the FRR, which is by definition the EER. It could be seen that in all cases, a slight change in FAR will lead to a greater change in FRR. In other words, the system will have a maximum FAR regardless of how "loose" the decision threshold was. Table 5-15 to Table 5-17 provide the FRRs (%) for different ideal FARs (%), for each case. In all situations, the FRRs increased greatly if the FAR was limited to 0.1% from 0.5%, especially when testing with babble noise. For a system which is not too strict on false acceptance, 0.5% of FAR will give an acceptable FRR.

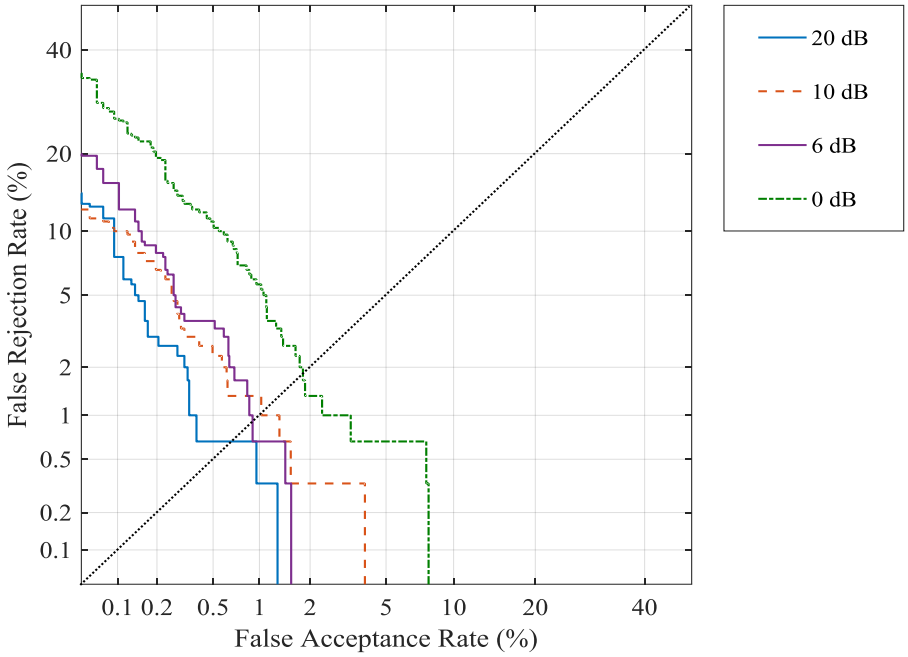

Figure 5-9: DET curves of i-vector/PLDA with compensation under babble noise

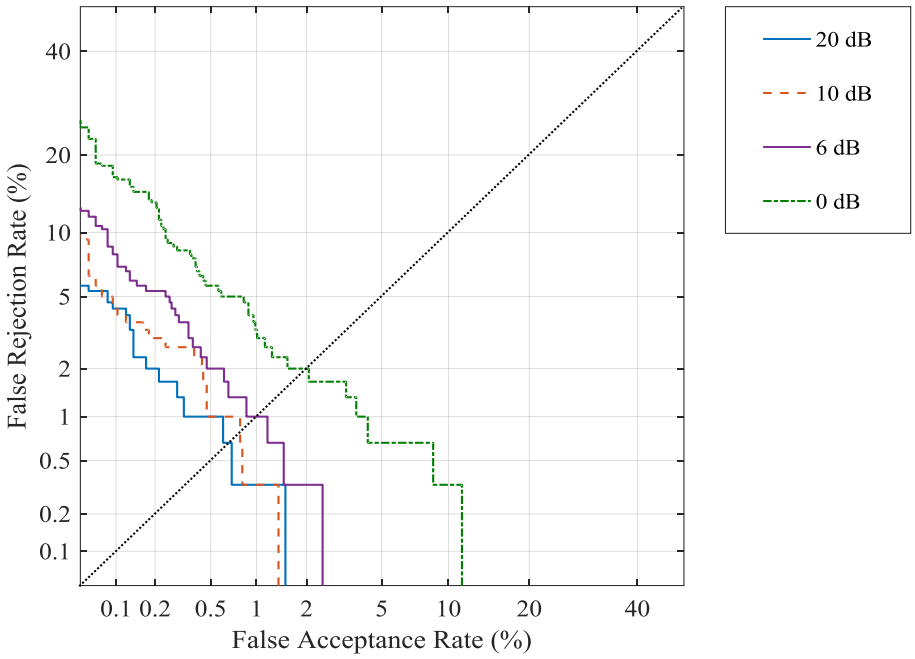

Figure 5-10: DET curves of i-vector/PLDA with compensation under car noise

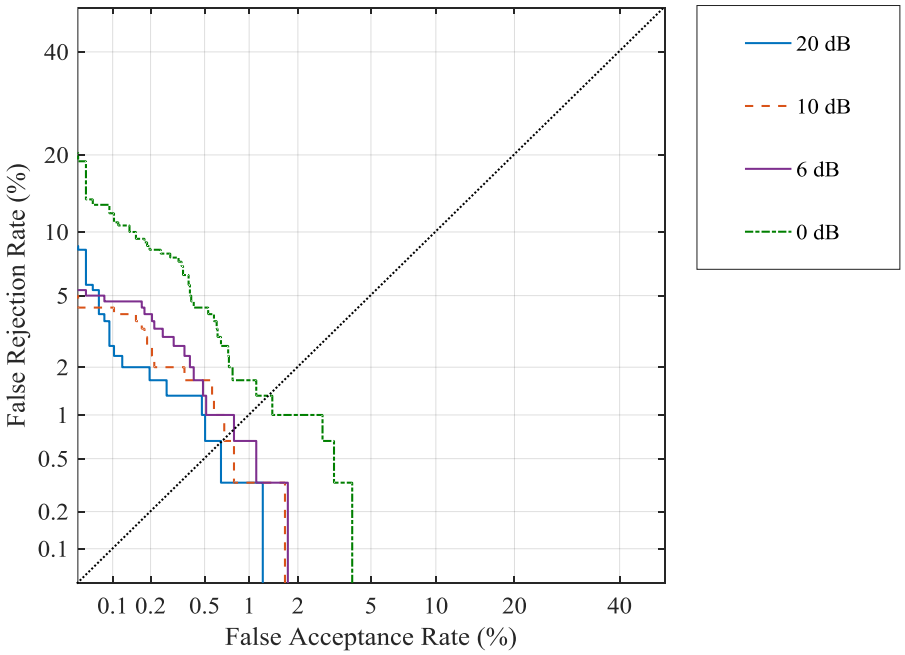

Figure 5-11: DET curves of i-vector/PLDA with compensation under office noise

Table 5-15: FRRs (%) of i-vector/PLDA with compensation under babble noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**0.33**	**1.33**	**0.67**	5.67
0.5	**0.67**	**2.33**	**3.67**	11.00
0.1	7.67	10.00	15.67	26.00

Table 5-16: FRRs (%) of i-vector/PLDA with compensation under car noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**0.33**	**0.33**	**1.00**	**3.33**
0.5	**1.00**	**1.00**	**2.00**	5.67
0.1	**4.33**	**4.33**	8.00	16.67

Table 5-17: FRRs (%) of i-vector/PLDA with compensation under office noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**0.33**	**0.33**	**0.67**	**1.67**
0.5	**1.00**	**1.67**	**1.33**	**4.33**
0.1	**2.67**	**4.33**	**4.67**	12.00

5.6.2 Unanticipated Airplane Noise

The results obtained from S4 and S5 fusion with unanticipated airplane noise are shown in Figure 5-12 and Table 5-18 and Table 5-19. As in the previous section, small changes in FAR will lead to bigger changes in FRR. Considering the fact of previously unseen noise, the FRRs became extremely large when the FAR was limited to 0.1%, especially in low SNR situations. If a system requires low FAR (for example 0.1%), the FRR can be as high as 30%. For a system that can accept 0.5% false acceptance, FRR will be less than 10%; if a system can accept 1% false acceptance, then FRR can be less than 6%.

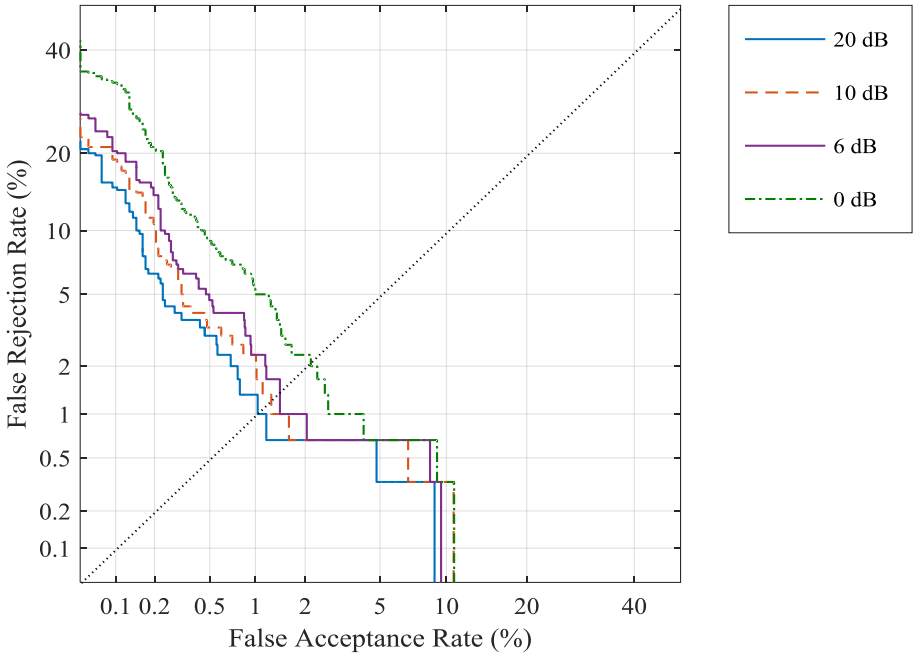

Figure 5-12: DET curves of S4 under airplane noise

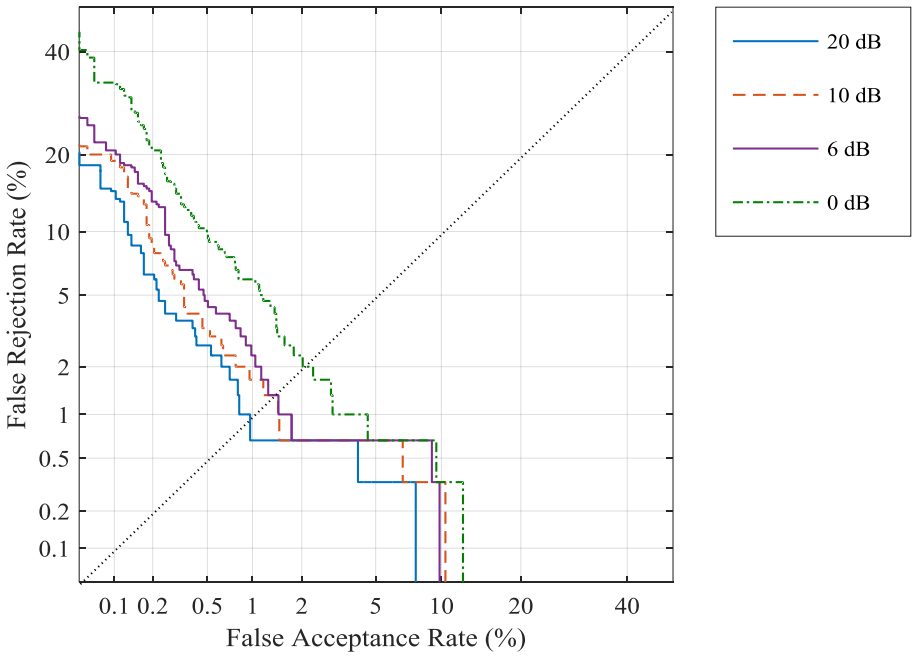

Figure 5-13: DET curves of S5 under airplane noise

Table 5-18: FRRs (%) of S4 under airplane noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**1.33**	**2.33**	**2.33**	5.33
0.5	**3.00**	**3.33**	**4.67**	9.00
0.1	15.00	19.00	20.33	33.00

Table 5-19: FRRs (%) of S5 under airplane noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**0.67**	**1.67**	**2.33**	6.00
0.5	**2.67**	**3.33**	**4.67**	10.00
0.1	14.67	20.67	19.00	33.33

Chapter 6 C++ Implementation for Real Time Online Processing

6.1 System Description

Based on the experimental results obtained through MATLAB tests in the previous chapter, it can be concluded that for a small-set speaker verification system that works under multi-SNR and multi-noise conditions, using 60-dim MFCCs features (19 MFCCs plus energy, along with deltas and double-deltas) combined with the i-vector/PLDA with compensation approach is a good solution. The i-vector/PLDA with compensation method gave a stable performance for both previously seen and previously unseen noises, meanwhile the i-vector/Cosine approach was not found to be as robust when testing with previously seen noises. Although the fusion systems S4 and S5 were able to improve the performance under previously unseen airplane noise, they require the computation of two independent systems. Hence, a C++ based speaker verification system using the ivector/PLDA with compensation approach is implemented and described in this chapter. For example, this system can work as a continuously working voice lock or intelligent doorbell application for a small-set family. It can also be seen as a surveillance system which could report strangers detected in a house.

There are two vital features of this system: online processing and multi-threading. Since the application should work constantly, an online processing technique was used. We assume that the system receives audio segments every ten milliseconds, while we have previously determined that a reliable speaker verification decision should be made based on at least 5-8 sec. of speech. Here we will use 8 sec. for the speech utterances length. Instead of storing all the short audio segments in memory until they comprise a long sentence, the system will continuously extract MFCCs using the newly received samples and it will store those features in buffer. The main advantage of this approach is not necessarily to save storage space, but it is to re-use each computed frame of MFCCs for several classification decisions (over 8 sec.). Since we use 25 ms windows to compute the MFCC features, and since we receive new speech frames (of 10 ms) every 10ms, it is natural to view our system as using 25 ms windows with 15 ms overlap. For every new 8 sec. of speech, a frame update every 10 ms means that 800 new frames of MFCC are computed. Note that for this system to be able to run in real-time, the computation of a frame of MFCC features must

take less than 10 ms (and preferably much less than 10 ms, since some of the processing power must also be left available for all the other computations to be performed by the overall system). Only the 19-dim original coefficients and their energy are extracted during this stage, resulting in 800×20 feature vectors in a buffer. Note that the buffer is not filled all at once (e.g. offline) for the whole 8s duration, instead it is continuously filled with a new set of MFCC coefficients every time that a new frame of speech data is received, i.e., every 10 ms. This can be achieved by performing a shift operation on the MFCC frames buffer, discarding the oldest MFCC coefficients and then inserting the new MFCC coefficients. Or more efficiently using a "circular buffer" approach, where the position of the current/latest MFCC coefficients is stored in a variable.

Once the buffer is filled with feature vectors (after the initial first 8 s), the MFCC features in the buffer can be processed to make verification decisions. This process includes adding deltas and double deltas coefficients, extracting i-vectors, applying compensation strategies and PLDA scoring. From our implementation, we have determined that this process takes about 500 milliseconds in total, on a recent general purpose computer. However, we do not want our system to make a new verification decision every time that a new frame of MFCC coefficients is computed (i.e., every 10 ms), and in fact this would not be possible in real time since the computation takes around 500 ms. Considering that continuously processing the recorded signal will lead to some non-speech segments and partial-speech segments, and also considering the complexity of the resulting system, we have decided that a speaker verification decision will be made every 3.2 sec., i.e., whenever 320 new MFCC frames have been saved in the buffer. Thus the buffer to be filled with new MFCC frames will be of size 320×20, while a longer buffer of size 800×20 (including the 320×20 buffer) will be used to process the 8 sec. of data and make a speaker verification decision. While the computations required for a verification decision are made, the system will also simultaneously receive new audio segments every 10 ms. Thus two different threads are used to run the MFCC feature extraction and the speaker verification processing separately (which we refer to as the short thread and the long thread, respectively). In such case, the system is able to read the 10ms audio segments and extract the MFCCs from it continuously, even while making a verification decision. More specifically, both threads are put to sleep most of the time, while the short thread is waken up every 10 ms and the long thread is notified as soon as the 320×20 MFCC

feature buffer is full (every 3.2 sec.). Both threads go back to sleep once they finish their job. The working system is illustrated in Figure 6-1.

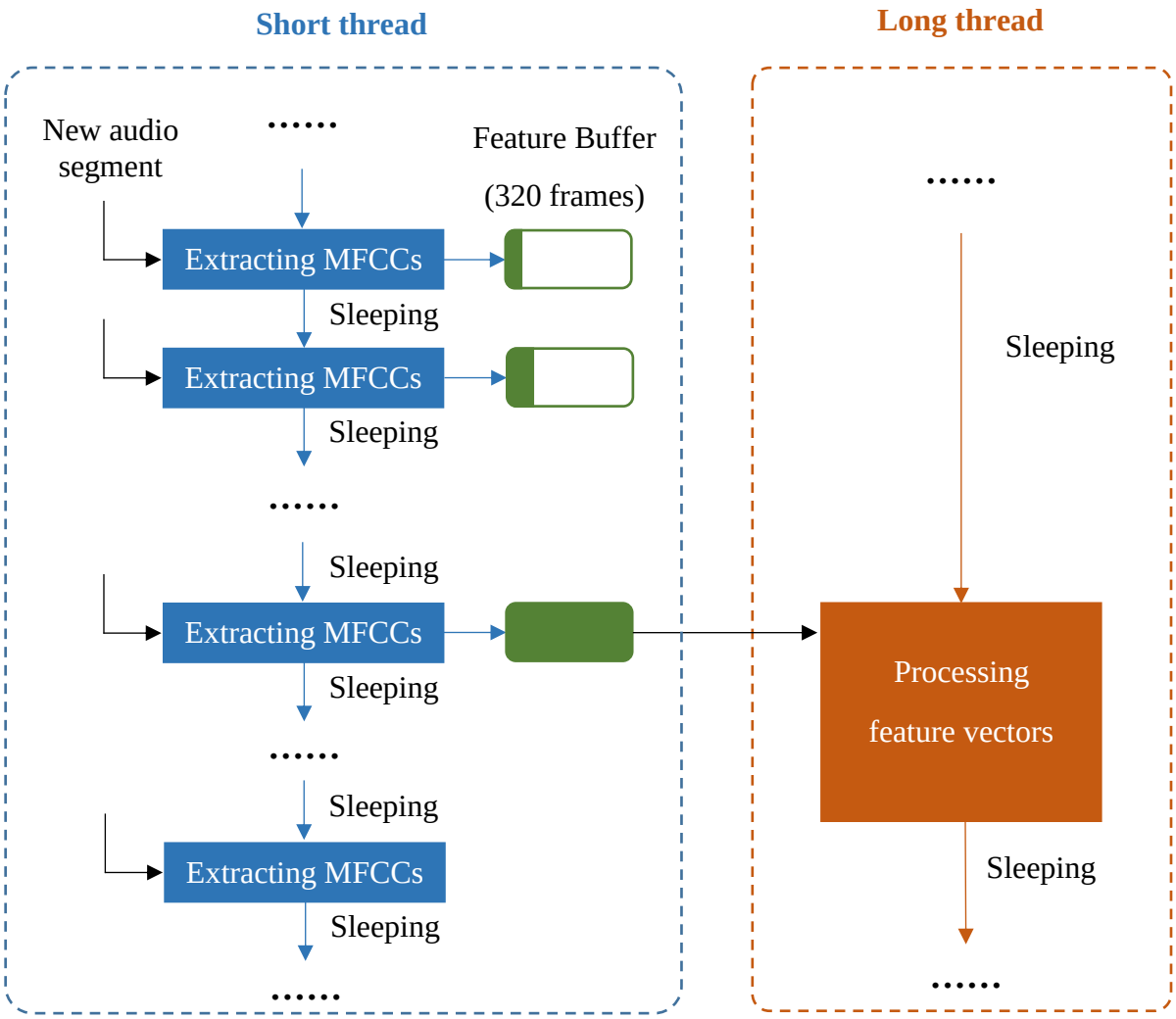

Figure 6-1: Workflow of C++ online system

6.2 Experimental Results

6.2.1 Without Online Processing

This section is to evaluate the performance of the offline C++ implementation of the i-vector/PLDA approach, and the same data and experimental setup as in section 5.3 were used. This means that in this section speech files were read "all at once" and there was no frame by

frame online processing, which is considered in the next section. The EERs obtained under the four noises and four SNRs are shown in Table 6-1. Although similar trends can be observed, the performance was not entirely as robust as the previous one in section 5.3 obtained from the MATLAB experiments. Compared to the results in Table 5-8, for the scenarios with the three previously seen noises, the averaged EER decreased by 0.08% and 0.12% for the 20 dB and 10 dB SNR cases, while the EER increased by 0.25% and 0.36% at 6 dB and 0 dB SNR. As for the unanticipated airplane noise, the C++ implementation results were also outperformed by the previous MATLAB simulation results in most cases, with EERs 1.2 and 1.5 times greater than in Table 5-9 at 6 dB and 0 dB SNR. Since the same data was used for the C++ and Matlab experiments, the differences of performance are essentially caused by the use of different libraries and toolboxes for the processing, as well as possibly different numerical resolutions. The overall performance of the C++ implementation is nevertheless still acceptable, with less than 1% EER when the testing speech utterances were relatively clean (20 dB and 10 dB SNR) and nearly 1% and 3% EER at 6 dB and 0 dB SNR, respectively, for the cases of previously seen noises. Although the EER obtained at 0 dB with airplane noise is 6.33% (above 5%), it is still acceptable since an unknown noise at 0 dB is representative of an extreme case.

Table 6-1: EERs (%) of offline C++ implementation

Noise type	**EER (%)**			
	20 dB	**10 dB**	**6 dB**	**0 dB**
Babble	0.67	1.04	1.55	2.69
Car	0.33	0.47	0.75	1.84
Office	0.31	0.34	0.77	2.00
Averaged across babble, car and office	0.44	0.62	1.02	2.18
Airplane (unseen)	1.67	3.39	4.16	6.33

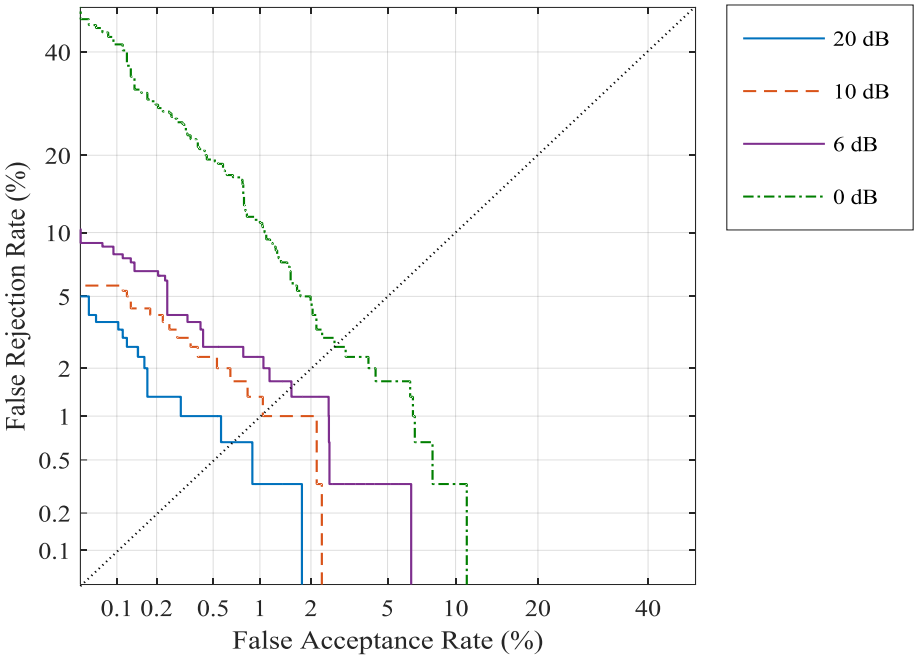

Figure 6-2: DET curves of offline C++ implementation under babble noise

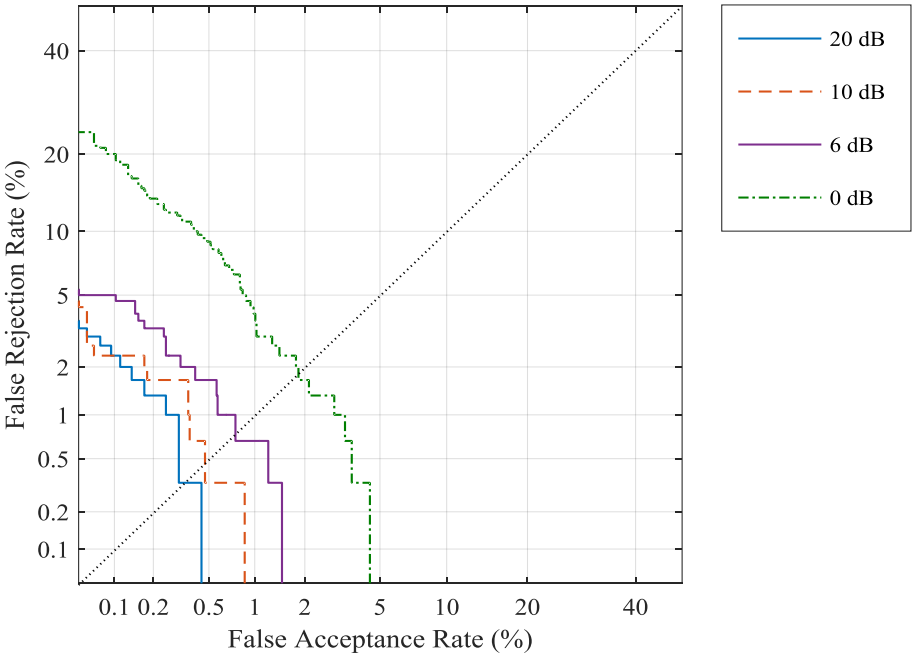

Figure 6-3: DET curves of offline C++ implementation under car noise

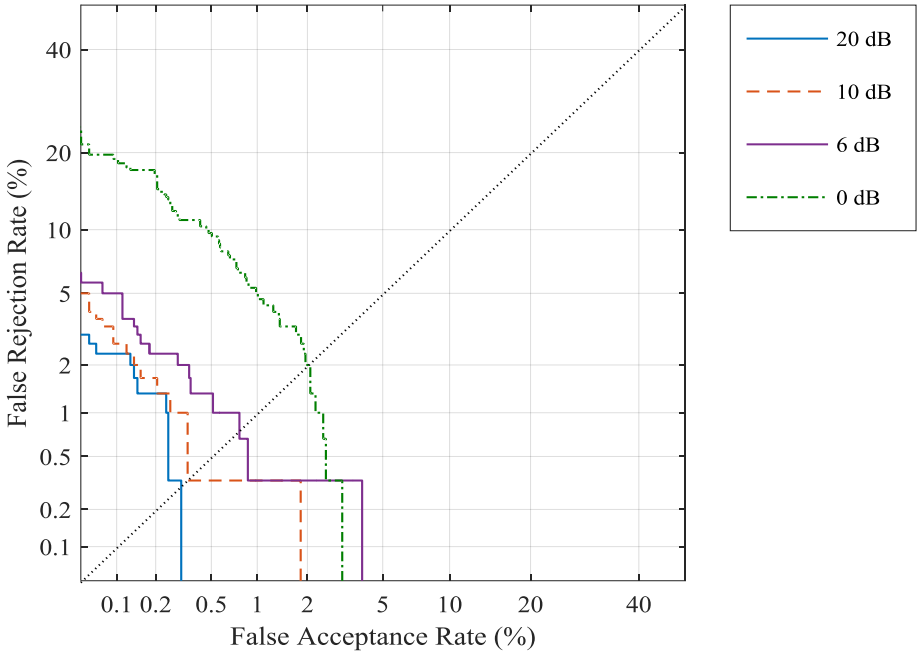

Figure 6-4: DET curves of offline C++ implementation under office noise

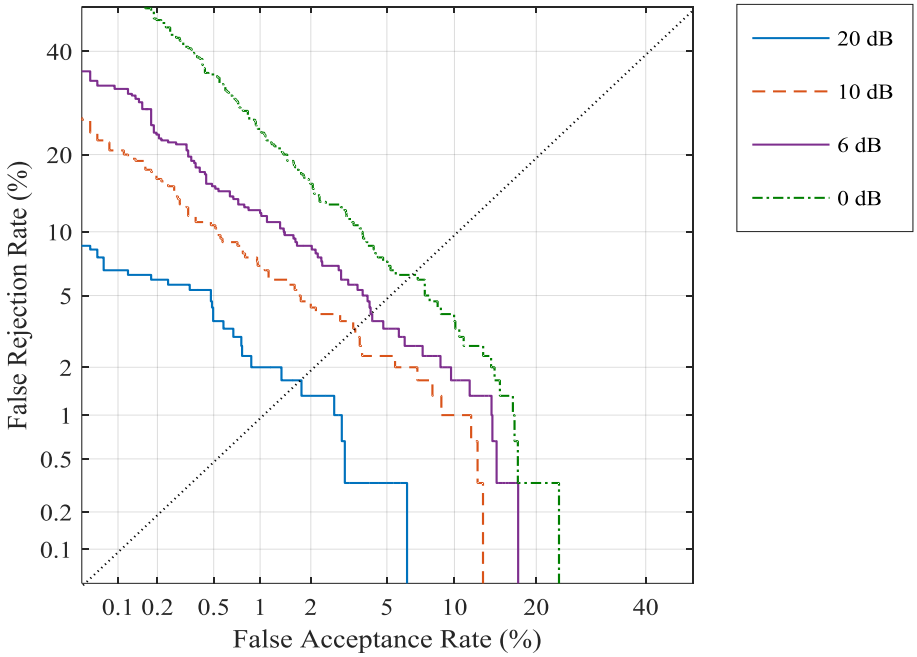

Figure 6-5: DET curves of offline C++ implementation under airplane noise

Table 6-2: FRRs (%) of offline C++ implementation under babble noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**0.33**	**1.33**	**2.33**	11.33
0.5	**1.00**	**2.33**	**2.67**	19.33
0.1	**3.67**	5.67	8.00	41.67

Table 6-3: FRRs (%) of offline C++ implementation under car noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**0.00**	**0.00**	**0.67**	**4.00**
0.5	**0.00**	**0.33**	**1.67**	9.00
0.1	**2.33**	**2.33**	**5.00**	20.00

Table 6-4: FRRs (%) of offline C++ implementation under office noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**0.00**	**0.33**	**0.33**	**5.00**
0.5	**0.00**	**0.33**	**1.33**	9.67
0.1	**2.33**	**2.67**	**5.00**	19.00

Table 6-5: FRRs (%) of offline C++ implementation under airplane noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**2.00**	7.00	12.33	24.33
0.5	**3.67**	10.67	15.33	35.00
0.1	6.67	20.67	32.00	53.67

The DET curves for the four noise conditions are shown in Figure 6-2 to Figure 6-5, meanwhile Table 6-2 to Table 6-5 present the corresponding FRRs with different FARs. For previously anticipated noises, almost all FFRs were smaller than the ones in Table 5-15 to Table 5-17 (Matlab experiments) when the SNR was higher than 0 dB, but the C++ setup gave higher FRR when the SNR drops to 0 dB. As for the previously unseen airplane noise, the FRRs

are significantly larger than the ones with other noises. The FRRs are less than 10% only for the 20 dB SNR case and 10 dB SNR case, with a 1% FAR. The FRR is over 59% in the worst case testing with 0 dB SNR speech along with 0.1% FAR. In other words, in this case a target speaker will need to make at least two attempts (on average) to get approved.

6.2.2 Online Processing

In order to evaluate the online processing scheme, a different testing dataset was created. Instead of individual 11 sec. speech utterances from speakers, longer audio signals are more suitable to reproduce continuous processing conditions. Therefore the 400 speech utterances (11 sec. each) from 40 speakers (30 target speakers plus 10 impostors) were combined with babble, car, office and airplane noises, and the speech utterances were then concatenated in random order with 10 to 20 sec. of pure noise (same type as the one added to speech) between 2 speech utterances. In order to examine the system performance against different SNR condition, noises were again added at 20 dB, 10 dB, 6 dB and 0 dB separately. More specifically, 4 types of noise along with 4 SNR situations resulted in 16 long audio signals, and only one type of noise was included in each setup. As a result, approximately 170-minutes of audio was generated for each case. An example of 120 sec. of a long speech is shown in Figure 6-6, where speakers and noises are marked in different colours.

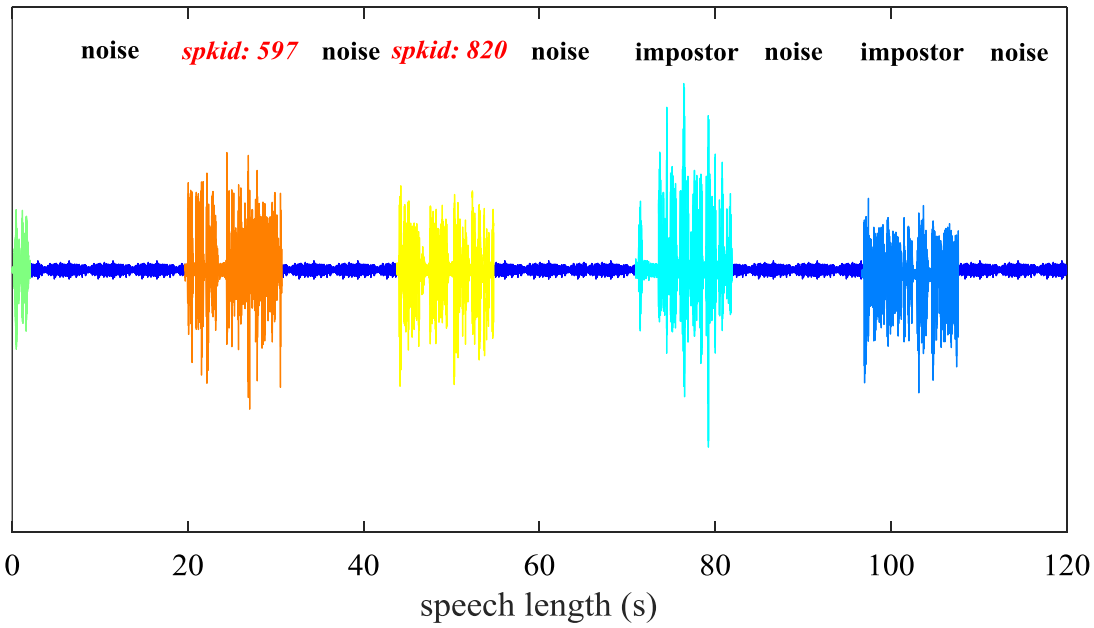

Figure 6-6: An example of a combined long speech

Considering the idea of "online processing", in addition to the target speech and the impostor 8 sec. segments, there would be segments that cover both target speech and pure noise, or both impostor speech and pure noise, as well as noise-only segments. A segment that contains both target speech and pure noise can be categorized as both positive and negative. Therefore, the following scheme for labelling was applied:

1. During the process of generating the long signal, the audio was first labelled sample by sample.
2. Segment the sample by sample labels the same way as online processing, with 800 frames window and 320 frames window shifts.
3. For each label segment, if there is one type of label (speaker id or noise) that covers over 90% of the whole segment, then the segment will be labelled after that type. Otherwise the segment as well as the corresponding score segment will be excluded. In other words, the segment will not be considered when evaluating the system performance since it can be both positive and negative.

Based on the above labelling strategy, the results obtained through the online systems are presented in Table 6-6. It can be noticed that the results are improved compared to Table 6-1, this

is caused by the noise-only segments included in this setup. Pure noise segments are easier for the system to distinguish from target speakers than impostors, therefore the overall EERs became less. Another observable performance is that in all cases, the EERs grow in a slow pace when the SNR drops from 20 dB to 6 dB but increase rapidly when the SNR reaches 0 dB. The averaged-EERs for the 0 dB SNR case are around 4 times worse than for 6 dB SNR, except for the previously unseen airplane noise, where the overall results are also better than the ones in Table 6-1, especially when the testing speech has 0 dB SNR.

Table 6-6: EERs (%) of online C++ processing

Noise type	**EER (%)**			
	20 dB	**10 dB**	**6 dB**	**0 dB**
Babble	0.18	0.23	0.26	1.12
Car	0.47	0.39	0.58	1.40
Office	0.11	0.21	0.24	1.04
Averaged across babble, car and office	0.25	0.28	0.36	1.19
Airplane (unseen)	0.62	1.98	3.16	3.99

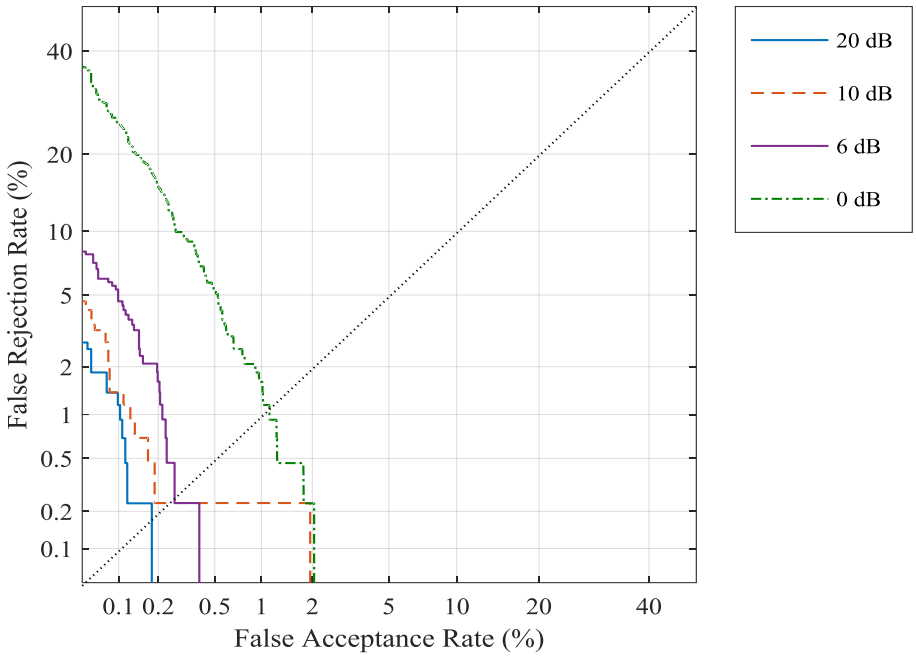

Figure 6-7: DET curves of online C++ processing under babble noise

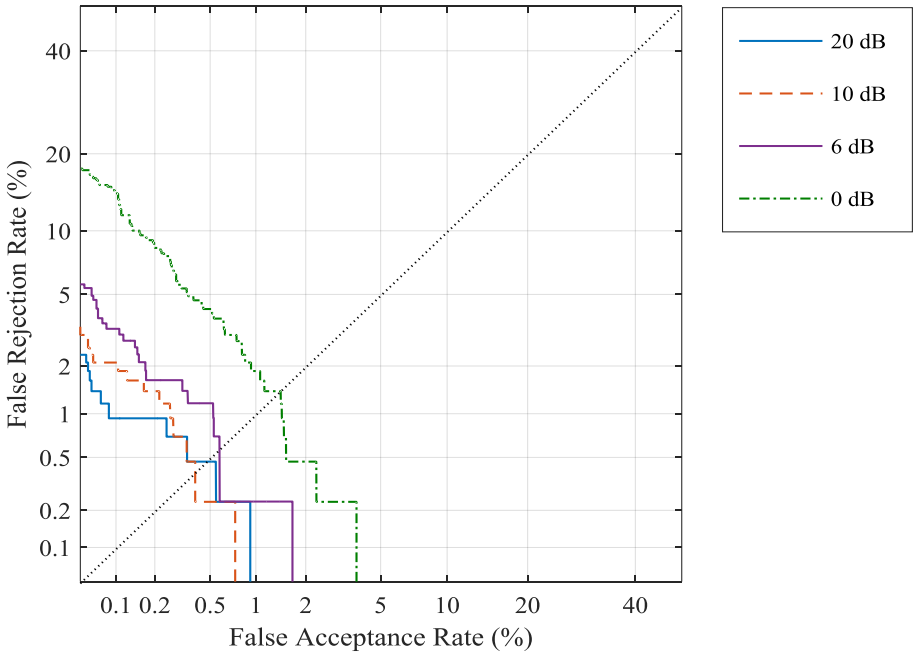

Figure 6-8: DET curves of online C++ processing under car noise

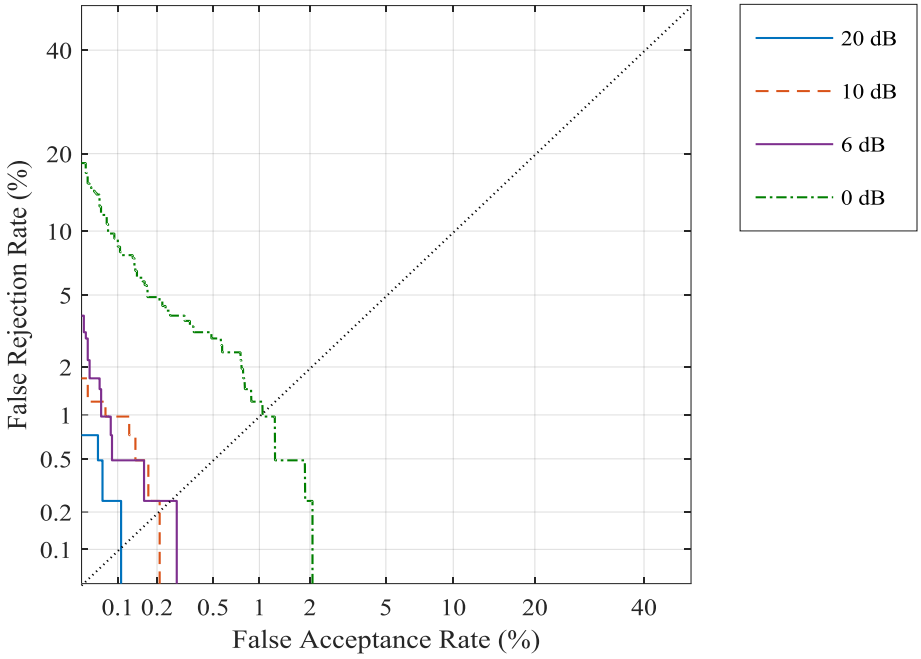

Figure 6-9: DET curves of online C++ processing under office noise

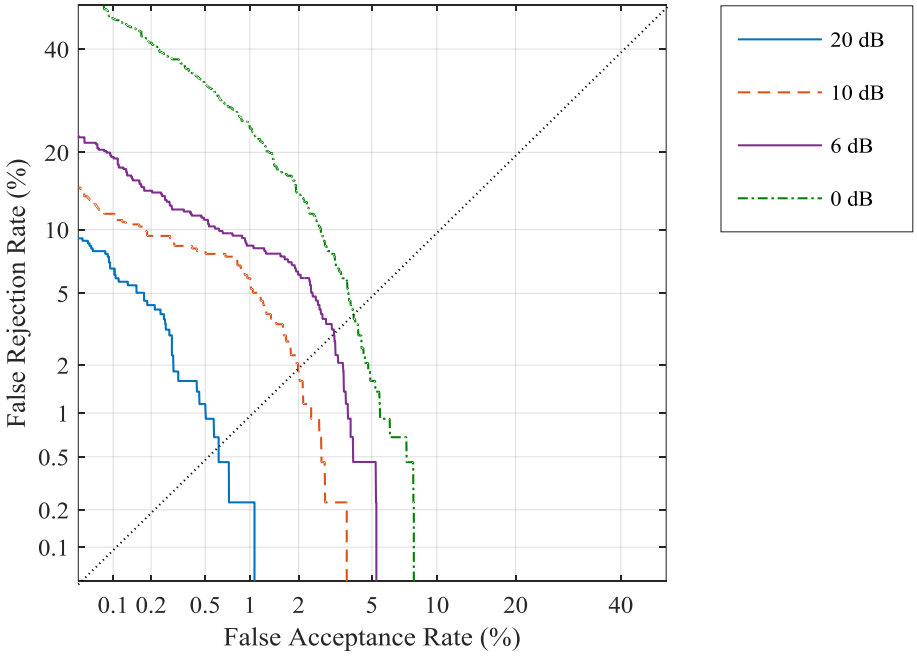

Figure 6-10: DET curves of online C++ processing under airplane noise

Table 6-7: FRRs (%) of online C++ processing under babble noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**0**	**0.23**	**0**	**1.62**
0.5	**0**	**0.23**	**0**	5.31
0.1	**1.15**	**1.39**	**4.64**	25.17

Table 6-8: FRRs (%) of online C++ processing under car noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**0**	**0**	**0.23**	**1.86**
0.5	**0.23**	**0.47**	**1.17**	**4.20**
0.1	**0.93**	**2.10**	**3.28**	14.45

Table 6-9: FRRs (%) of online C++ processing under office noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**0**	**0**	**0**	**1.22**
0.5	**0**	**0**	**0**	**2.93**
0.1	**0.24**	**0.98**	**0.48**	9.27

Table 6-10: FRRs (%) of online C++ processing under airplane noise

FAR (%)	FRR (%)			
	20 dB	10 dB	6 dB	0 dB
1.0	**0.23**	**5.95**	8.49	24.03
0.5	**1.14**	8.01	11.01	32.49
0.1	6.64	11.67	19.27	46.91

The DET curves and FRRs of the online processing system are given in Figure 6-7 to Figure 6-10 and Table 6-7 to Table 6-10. The results are better compared to the ones in Chapter 5 and in the previous offline setup, especially when FARs are limited to 0.1% at high SNR conditions. And the improvements become smaller when testing with 0 dB SNR noisy speech. This can be explained the same way as in the previous section: it is easier to distinguish pure noise

from target speech than impostor speech from target speech. For previously seen noises, 0.5% FAR for this system will give low FRR in most cases, except for 0 dB SNR conditions. Even if the system has high security standards and only allows 0.1% FAR, the FRR will still be less than 2% in clean situations with previously anticipated noises. However, like in the offline experiments, the situation becomes more difficult for previously unseen airplane noise: 0.5% FAR will lead to 11.01% and 32.49% FRRs for the 6 dB SNR and 0 dB SNR cases. Therefore, there will be a high chance for the system to give a rejection if a target speech is processed under a new type of loud noise.

Chapter 7 Conclusion

The thesis aimed at evaluating various speaker verification methods in multi-noise and multi-SNR conditions, meanwhile finding the best solution for a small-set speaker verification system in practice. Unlike the commonly used experimental setup in other studies, where training speech is at the same SNR and noise distribution as testing speech, training speech in this work was under a nearly uniform SNR distribution. Therefore, there is great mismatch between training and testing speech, which forms a more practical situation. Besides, an unknown noise which was not included in training speech was used for testing the system performance. Moreover, this thesis analyzed the impact of the testing speech length on the verification performance, in order to get the shortest testing length that will give a reliable result. At last, a C++ online processing system was implemented, based on the previous experiments.

Chapter 5 presented the experiments of different methods as well as different testing cases. The i-vector/PLDA with compensation method was found to be the most robust approach when testing with previously anticipated noises, while the i-vector/SI-mPLDA and the i-vector/SD-mPLDA methods also achieved acceptable results. In the experiments with a previously unseen airplane noise, the i-vector/Cosine method showed a superiority at low SNR conditions while the i-vector/PLDA with compensation approach also gave an acceptable performance. With previously seen noises, the fusion systems of conventional PLDA and mixture of PLDA did not bring observable improvement. Also, the fusion systems require computing several algorithms, therefore it is not necessary to perform linear fusion in this case. For previously unseen noise, the fusion system of i-vector/Cosine and i-vector/PLDA with compensation methods improved the EER performance by 1.5 to 2 times, thus it is a good solution for this setup.

The experiments for the testing speech length showed that 8 sec. speech utterances can achieve similar results as 11 sec. speech at high SNR conditions and the system performance degraded rapidly when testing speech became shorter than 5 sec. Therefore, in high SNR cases a speech length longer than 5 sec. will lead to good verification results. However, the speech length needs to be longer when the speech SNR is lower or with a previously unseen noise. With respect to the analysis on FRR and FAR, for both previously seen and unseen noises, the best systems can

achieve less than 5% FRRs with SNR higher than 0 dB when the FAR is larger than 0.5%. And the false rejection rate will grow rapidly if the testing SNR drops to 0 dB or if the FAR is set to 0.1%.

The C++ implementation achieved similar results as the MATLAB experiments when testing with previously seen noises, while the C++ results were outperformed by MATLAB simulation with previously unseen airplane noise at low SNR situations. The overall performance of the C++ implementation working offline was found to be acceptable. The online processing gave better results regarding the EER compared to the offline system, this is mainly because there are noise-only segments in the online tests, which are easier for the system to distinguish from target speech.

There are some limitations to the scope of this thesis. The types of noises are limited, and more complex setups with more noise types, room reverberation and different recording devices can be considered. Also, this thesis did not consider the case of speakers trying to imitate other people's voice by intention. Besides, there is some more future work that could be done for this thesis. First, the combination of different types of features could be tested, as it could be a way to improve system performance but at the cost of increased computation complexity and storage. Second, the impact of the training speech length on the verification performance should also be investigated in more depth. Although it may not be as important as the testing speech length since the training is completed offline, it is still useful to know the best length of training speech that will lead to good results. Third, with the growing popularity and success of deep learning in the machine learning area, algorithms using neural networks could also be further applied for speaker verification. However, a larger amount of data as well as more training time would be required in this case. The current C++ implementation does not include fusion of speaker verification systems for computational simplicity, and an implementation of fusion system S5 could be developed if the requirement on the system complexity is not very strict. At last, a multimodal fusion of the speaker verification system could be performed with the classification obtained from an image/video processing system, to further improve the performance of the system.

References

Auckenthaler, R., Carey, M., & Lloyd-Thomas, H. (2000). Score Normalization for Text-Independent Speaker Verification Systems. *Digital Signal Processing*, *10*(1), 42–54.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

Campbell, W. M., Sturim, D. E., & Reynolds, D. A. (2006). Support vector machines using GMM supervectors for speaker verification. *IEEE Signal Processing Letters*, *13*(5), 308–311.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Davis, S., & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *28*(4), 357–366.

Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., & Ouellet, P. (2011). Front-End Factor Analysis for Speaker Verification. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(4), 788–798.

Dehak, Najim. (2009, June 23). *Discriminative and generative approaches for long- and short-term speaker characteristics modeling : application to speaker verification* (phd). École de technologie supérieure, Montréal.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, *39*(1), 1–38.

Fazel, A., & Chakrabartty, S. (2011). An Overview of Statistical Pattern Recognition Techniques for Speaker Verification. *IEEE Circuits and Systems Magazine*, *11*(2), 62–81.

Freesound.org - Freesound.org. (n.d.). https://www.freesound.org/

Furui, S. (1981). Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *29*(2), 254–272.

Furui, S. (1986). Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *34*(1), 52–59.

Garcia-Romero, D., Zhou, X., & Espy-Wilson, C. Y. (2012). Multicondition training of Gaussian PLDA models in i-vector space for noise and reverberation robust speaker recognition. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* 4257–4260.

Hansen, J. H. L., & Hasan, T. (2015). Speaker Recognition by Machines and Humans: A tutorial review. *IEEE Signal Processing Magazine*, *32*(6), 74–99.

Hatch, A. O., Kajarekar, S. S., & Stolcke, A. (2006). Within-class covariance normalization for SVM-based speaker recognition. Presented at the INTERSPEECH 2006 - ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA.

Hermansky, H., & Cox, L. A. (1991). Perceptual Linear Predictive (PLP) Analysis-Resynthesis Technique. In *Final Program and Paper Summaries 1991 IEEE ASSP Workshop on Applications of Signal Processing to Audio and Acoustics*, 37–38.

Hermansky, H., & Morgan, N. (1994). RASTA processing of speech. *IEEE Transactions on Speech and Audio Processing*, *2*(4), 578–589.

Hermansky, Hynek. (1990). Perceptual linear predictive (PLP) analysis of speech. *The Journal of the Acoustical Society of America*, *87.4*, 1738–1752.

Jackson, L. B. (1996). Filter Design by Modeling. In *Digital Filters and Signal Processing*. Springer US, 323–372.

Jain, A. K., Ross, A., & Prabhakar, S. (2004). An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, *14*(1), 4–20.

Joint Factor Analysis Matlab Demo | Speech Processing Group. (n.d.). http://speech.fit.vutbr.cz/en/software/joint-factor-analysis-matlab-demo

Kaldi ASR. (n.d.). http://kaldi-asr.org/

Kenny, P. (2005). *Joint factor analysis of speaker and session variability: Theory and algorithms* (No. CRIM-06/08-13). Montreal Quebec Canada: CRIM.

Kenny, P. (n.d.). Bayesian speaker verification with heavy tailed priors. The Speaker and Language Recognition Workshop (Odyssey 2010). Brno, Czech Republic.

Kenny, P., Boulianne, G., Ouellet, P., & Dumouchel, P. (2007). Joint Factor Analysis Versus Eigenchannels in Speaker Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, *15*(4), 1435–1447.

Kenny, P., & Dumouchel, P. (2004a). Disentangling speaker and channel effects in speaker verification. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 37-40.

Kenny, P., & Dumouchel, P. (2004b). Experiments in Speaker Verification using Factor Analysis Likelihood Ratios. In *Proc. Odyssey*.

Kenny, P., Mihoubi, M., & Dumouchel, P. (n.d.). New MAP estimators for speaker recognition. *Proceedings of the 8th European Conference on Speech Communication and Technology (Eurospeech 2003)*, 2964–2967.

Kenny, P. (n.d.). Bayesian speaker verification with heavy tailed priors. The Speaker and Language Recognition Workshop (Odyssey 2010). Brno, Czech Republic.

Lei, Y., Burget, L., Ferrer, L., Graciarena, M., & Scheffer, N. (2012). Towards noise-robust speaker recognition using probabilistic linear discriminant analysis. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4253–4256

Lyu, S., & Simoncelli, E. P. (2009). Nonlinear Extraction of Independent Components of Natural Images Using Radial Gaussianization. *Neural Computation*, *21*(6), 1485–1519.

Mak, M. W., Pang, X., & Chien, J. T. (2016a). Mixture of PLDA for Noise Robust I-Vector Speaker Verification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(1), 130–142.

Mak, M. W., Pang, X., & Chien, J. T. (2016b). Mixture of PLDA for Noise Robust I-Vector Speaker Verification. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *24*(1), 130–142.

Mak, M.-W. (2014). SNR-dependent mixture of PLDA for noise robust speaker verification. Presented at the Fifteenth Annual Conference of the International Speech Communication Association.

Makhoul, J. (1975). Linear prediction: A tutorial review. *Proceedings of the IEEE*, *63*(4), 561–580.

Mixture of PLDA. (n.d.). http://bioinfo.eie.polyu.edu.hk/mPLDA/

openslr.org. (n.d.). http://www.openslr.org/

Pang, X., & Mak, M.-W. (2015). Noise robust speaker verification via the fusion of SNR-independent and SNR-dependent PLDA. *International Journal of Speech Technology*, *18*(4), 633–648.

Piciarelli, C., Micheloni, C., & Foresti, G. L. (2008). Trajectory-Based Anomalous Event Detection. *IEEE Transactions on Circuits and Systems for Video Technology*, *18*(11), 1544–1554.

Prince, S. J. D., & Elder, J. H. (2007). Probabilistic Linear Discriminant Analysis for Inferences About Identity. In *2007 IEEE 11th International Conference on Computer Vision*, 1–8.

Rajan, P., Kinnunen, T., & Hautamaki, V. (2013). Effect of multicondition training on i-vector PLDA configurations for speaker recognition. *Proceedings of Interspeech*, 3694–3697.

Reynolds, D. A., & Rose, R. C. (1995). Robust text-independent speaker identification using Gaussian mixture speaker models. *IEEE Transactions on Speech and Audio Processing*, *3*(1), 72–83.

Reynolds, Douglas A, Quatieri, T. F., & Dunn, R. B. (2000). Speaker verification using adapted Gaussian mixture models. *Digital Signal Process*, *10*(1–3), 19–41.

Romero, D. G., & Wilson, E. (2011). Analysis of I-vector Length Normalization in Speaker Recognition Systems. *Proceedings of Interspeech*, 249–252.

Sadjadi, S. O., Slaney, M., & Heck, L. (2013). MSR Identity Toolbox v1.0: A MATLAB Toolbox for Speaker Recognition Research. *Microsoft Research*.

Stevens, S. S., Volkmann, J., & Newman, E. B. (1937). A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America*, *8*(3), 185–190.

Theodoridis, S., & Koutroumbas, K. (2009). *Pattern Recognition* (4th ed.). Academic Press.

VOICEBOX. (n.d.). https://www.mathworks.com/matlabcentral/linkexchange/links/797