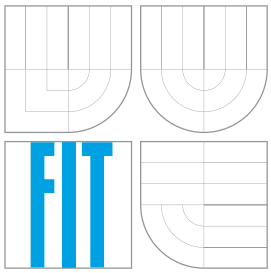
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ MLUVČÍHO NA MOBILNÍM TELEFONU

SPEAKER RECOGNITION ON MOBILE PHONE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JAN PEŠÁN
AUTHOR

VEDOUCÍ PRÁCE — Doc. Dr. Ing. JAN ČERNOCKÝ
SUPERVISOR

BRNO 2011

Abstrakt

Tato práce se zaměřuje na implementaci počítačového systému rozpoznávání řečníka do prostředí mobilního telefonu. Je zde popsán princip, funkce, a implementace rozpoznávače na mobilním telefonu Nokia N900.

Abstract

This work aims to port Speaker Identification System (SID) to the mobile device / mobile phone. We will describe basic principles, function and implementation of speaker identification system on Nokia N900 mobile phone.

Klíčová slova

Rozpoznání řečníka, Identifikace řečníka, Ověření řečníka, Nokia N900, ARM, iVector, PLDA, Biometrie

Keywords

Speaker recognition, Speaker identification, Speaker verification, Nokia N900, ARM, iVector, PLDA, Biometry

Citace

Jan Pešán: Speaker Recognition on Mobile Phone, diplomová práce, Brno, FIT VUT v Brně, 2011

Speaker Recognition on Mobile Phone

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc. Dr. Ing. Jana Černockého

.....
Jan Pešán
May 24, 2011

Poděkování

Rád bych na tomto místě poděkoval Všem lidem, bez kterých by tato práce nevznikla. Jsou to všichni členové výzkumné skupiny Speech@FIT, především pak Doc. Dr. Ing. Honza Černocký, Ing. Pavel Matějka, Ph.D., Ing. Ondřej Glembek a Dr. Lukáš Burget, Ph.D. Za technickou podporu musím též poděkovat Mikovi Helistekangas z firmy Visidon (FI), který mě provedl taji platformy ARM. Za perfektní organizaci celého projektu MOBIO pak týmu Sébastiena Marcela z výzkumného institutu IDIAP (CH). V neposlední řadě bych chtěl poděkovat těm, kteří mě po celou dobu mého studia neúnavně podporovali v mém snažení - mé rodině a mé přítelkyni.

Contents

Chapter 1

Introduction

If we look at the contemporary mobile phones, we can see that they are more than phones. They have turned into full-featured communication devices similar to personal computers. Everyone can use almost any application „on the go“, including online services. Mobile phones are able to bring us broadband internet, location services like GPS and help us in many other aspects of modern life.

However, we have to realize that mobile devices were not originally developed to provide security services. The PC provides us with a possibility to use USB tokens, fingerprint readers, Smartcard readers and almost every device that can serve for the purposes of personal verification and identification. There is of course a possibility to connect some of these peripherals to the mobile device, but it brings various problems from ergonomy to power consumption.

This work is a part of the MOBIO project, which aims to provide robust verification system for mobile phone users without need of the special hardware. It combines two different verification systems in order to bring better performance solely using frontal camera and microphone - Speaker- and Face-verification.

My work in this project consists of the implementation of the whole Speaker-verification system into the mobile phone. After successful implementation, I will measure performance of the system and optimize it for real-time processing. Last step is to measure quality of decisions produced by the system and compare them with baseline PC Speaker-verification system on the big data set.

In the Chapter 2, we will describe the basic Speaker recognition system with improved modeling part based on the JFA (Joint Factor Analysis) approach - iVector system. Chapter 3 is description of the MOBIO project, the partners in the project and target platform. Application frameworks and programming language are being dealt with in Chapter 4. Implementation of the speaker ID system to the mobile phone and its incorporation to the MOBIO framework is the main focus of Chapter 5. Because the whole system has to run on the mobile phone with limited resources, we had to make some performance optimizations described in Chapter 6. Finally, the elaboration on the evaluation of accuracy is presented in Chapter 7.

Chapter 2

Basis of Speaker Identification

Speaker identification systems can be created in various ways. In this project, we started with the baseline approach and then switched to the new *state-of-the-art* approach called iVector system.

2.1 Speaker Identification Basic Tasks

We can define two main problems of SID. In the the first approach to SID, we get two samples of unrestricted free speech and we want to determine whether these are recordings of the same speaker or of two different speakers. This problem definition is called *trial*. See Figure 2.1.

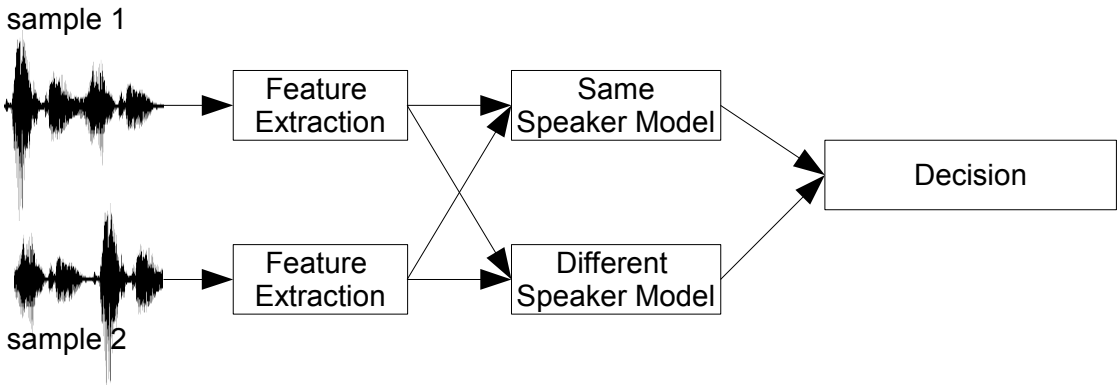


Figure 2.1: The First Approach to SID - Trial

The second approach is defined as two recordings: the first is the enrollment one, used to train speaker model, and the second is a testing one. We are asking the system whether the testing one is from the same speaker or not. This is the approach used in this work. See Figure 2.2.

2.2 Baseline SID

Baseline system scheme is in Figure 2.3. We used this system as basic system to derive our mobile implementation.

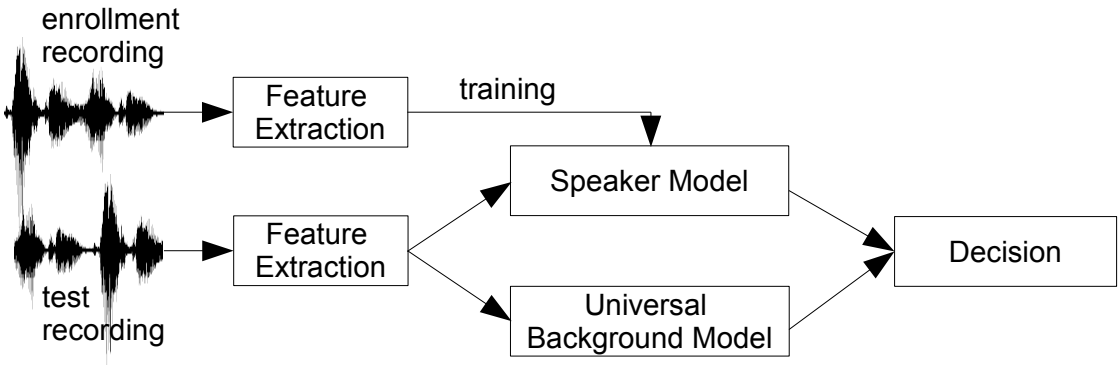


Figure 2.2: Approach to SID Used In This Work

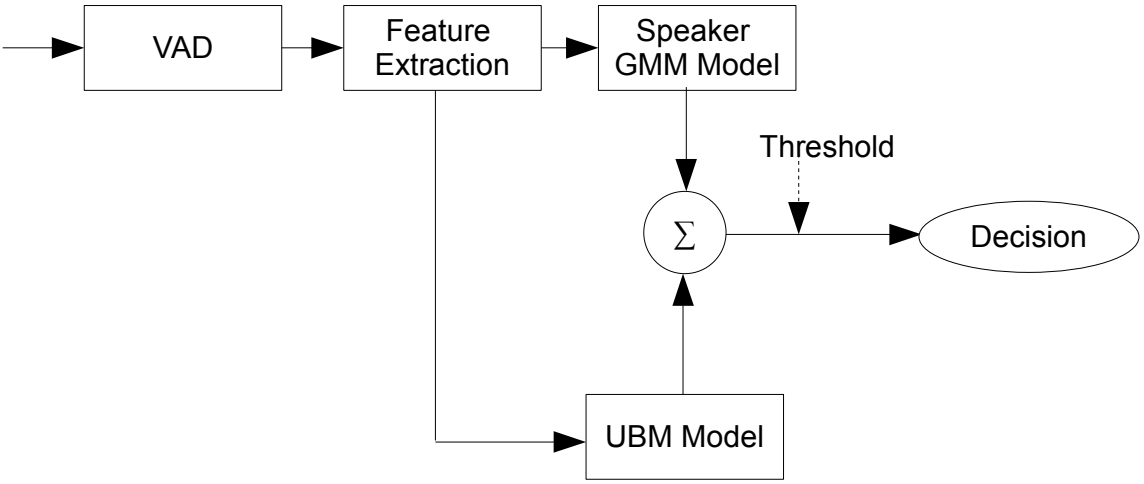


Figure 2.3: Scheme of the Baseline SID System without iVectors

2.2.1 VAD - Voice Activity Detection

For the proper function of Speech algorithms, it is necessary to know which part of audio data is related to speech and which part is only background noise. To achieve this goal, we can use many possible methods. For example, we can choose approaches which use Short-Time Energy for the detection of high energized phonemes, Zero-crossing rate, or more sophisticated approaches like computing Cross-correlation coefficients or using Neural nets and phoneme decoder. In this project, we used the last method - phoneme decoder. Details will be discussed later in Section 5.1.1.

2.2.2 Feature Extraction

Speech data can be described in various ways. It is naturally multidimensional description that consists of important features. There are nevertheless also many features which are unimportant for the purposes of speech processing. In case of SID, these include e.g. information about psychical status of the speaker, his actual health etc. Influence of these factors can be even undesirable, because it can confuse the system. Therefore, the extraction of the right features is very important for the success of the whole process. We used classical approach for speaker identification, consisting of the set of the following modules:

Melbank Filters

A reduced spectral representation is produced by passing the speech frame through logarithmically spaced filters with increasing bandwidths (mel-filters) designed to match the frequency sensitivity of the ear. The process of passing the speech frame through the mel-filters produce a spectral representation consisting of log magnitude values from the speech spectrum sampled at a logarithmic spacing [11].

Mel-Frequency Cepstral Coefficients

Then we create MFCCs by applying logarithm and DCT (Discrete Cosine Transformation). It gives us final vector of MFCC coefficients for each speech frame. The log magnitude spectral representation is then inverse Fourier transformed (in this case Inverse Discrete Cosine Transformation is used) to produce the final representation, called cepstral coefficients. DCT is used to decorrelate the log-magnitude spectrum samples [11].

Derivations

Derivation part approximates the first and the second order derivations of the MFCC trajectories.

Short Time Gaussianization

STG warps parameters locally using a floating window of fixed size (usually 3s long, but in our case shortened, see chapter 5.1.2). Parameters are warped with Inverse Gaussian cumulative density function to achieve locally Gaussian distribution, which is more suitable for GMM's.

2.2.3 Input Data

We are obtaining input data in the form of the MFCC coefficients vectors from the Feature extraction in fixed intervals. The whole utterance is sampled to the 30ms sliding windows with 10ms step between frames. It produces the sequence of extracted vectors denoted as $\mathcal{X} = \{\mathbf{x}_1 ... \mathbf{x}_T\}$ where T is number of frames.

2.2.4 Statistical Speaker Model - GMM

In this approach, we treat the speaker as a random source producing the observed feature vectors, as described in 2.2.2. Production of the concrete spectral vector feature is dependent on the state of the vocal tract. We can observe only the spectral parameters, internal configuration of the vocal tract which products them is hidden.

When we assume Gaussian distribution of observed feature vectors ($\mathcal{X}$), we can model posterior probability of one observed vector $\mathbf{x}_t$ by one Gaussian as [12]:

$$p(\mathbf{x}_t|c) = w_c \, \mathcal{N}(\mathbf{x}_t, \mu_c, \mathbf{\Sigma}_c), \tag{2.1}$$

where w is a weight and $\mu_c, \mathbf{\Sigma}_c$ are parameters of Gaussian c. We can generalize equation 2.1 for the whole model Θ (cohort of Gaussians) to the form:

$$p(\mathbf{x}_t|\mathbf{\Theta}) = \sum_{c=1}^{c} w_c \, \mathcal{N}(\mathbf{x}_t, \mu_c, \mathbf{\Sigma}_c). \tag{2.2}$$

If we want to model posterior probability of the whole sequence of feature vectors $\mathcal{X}$ by the model Θ we obtain the following equation:

$$p(\mathcal{X}|\Theta) = \prod_{t=1}^{T} \sum_{c=1}^{c} w_c \, \mathcal{N}(\mathbf{x}_t, \mu_c, \mathbf{\Sigma}_c). \tag{2.3}$$

If we want to obtain the occupation probability γ, we have to use the Bayesian formula: the likelihood of a frame for one Gaussian component is weighted by the prior probability of that class (Gaussian component weight w_c), and normalized by sum of such terms over all classes (Gaussians).

$$\gamma_t^{(c)} = \frac{w_c \, \mathcal{N}(\mathbf{x}_t, \mu_c, \mathbf{\Sigma}_c)}{\sum_{c=1}^{c} w_c \, \mathcal{N}(\mathbf{x}_t, \mu_c, \mathbf{\Sigma}_c)}. \tag{2.4}$$

Universal Background Model

Universal Background Model (UBM) [12] is a model trained on a large set of speakers. Internal structure of the model is the same as GMM model. It represents „neutral“ or „background“ speaker and we are using it in the following section 2.2.5 to evaluate the score of verification.

2.2.5 Scoring

In the basic paradigm, we are working with two hypotheses:

- h_0: speaker *is not* the person who trained the model
- h_1: speaker *is* the person who trained the model

Then we compute LLR (Log Likelihood Ratio) this way:

$$\Lambda(\mathcal{X}) = log\frac{\omega_1(\mathcal{X})}{\omega_0(\mathcal{X})} = log(p(\mathcal{X}|\Theta_{spk})) - log(p(\mathcal{X}|\Theta_{ubm})), \tag{2.5}$$

where ω_1 is a likelihood, that speech came from speaker model (it is produced by speaker model), and ω_0 is a likelihood, that speech *did not* come from speaker model (it is produced by UBM). Both ω_0 and ω_1 are computed using Equation 2.3.

This results in the LLR score, which is compared with tuned threshold θ. When $\Lambda > \theta$ speaker is confirmed, otherwise rejected.

2.3 iVector SID

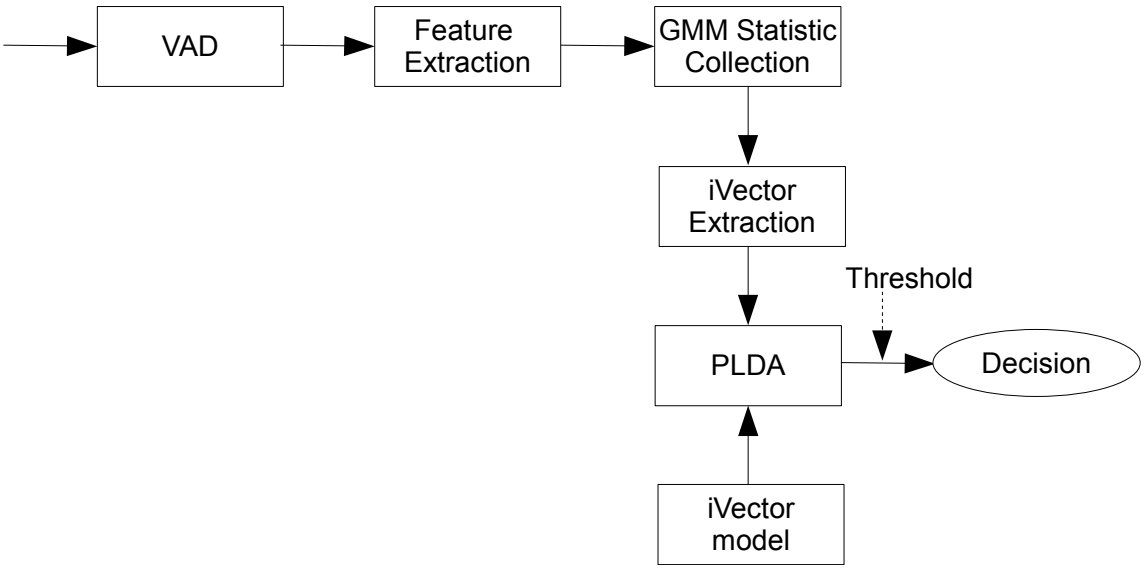


Figure 2.4: Scheme of the Modified SID System with iVectors

iVector system (or total variability system) is a new approach in speaker verification systems. It is important to be able to recognize speaker under various conditions. We can divide factors that affect performance of the speaker recognition into two categories. When we try to recognize the speaker, we can see that the performance of the recognition can be affected by many factors. We can divide this type of variability into the two classes:

- *intrinsic variability* - language, emotions, stress, Lombard effect, health, content of speech, etc.
- *extrinsic* - noise, microphone type, codec, recording medium, etc.

Together, these factors are called an *inter-session variability*. The other category is speaker variability. This variability describes difference between the speakers. It is obvious that we want to maximize speaker variability and minimize inter-session variability influence on the result of recognition. This concept is generalized and used in the Joint Factor Analysis as described in the next section.

2.3.1 JFA

In JFA, a supervector $\mathbf{M}$ that consists of stacked mean vectors of speaker- and session-specific GMM model represents a speaker utterance. Specifically, the speaker-dependent supervector is defined as

$$\mathbf{M} = \mathbf{m} + \mathbf{V}\mathbf{y} + \mathbf{U}\mathbf{x} + \mathbf{D}\mathbf{z}, \tag{2.6}$$

where $\mathbf{m}$ is a speaker- and session-independent supervector (generally from a Universal Background Model (UBM)), $\mathbf{V}$ and $\mathbf{D}$ define a speaker subspace (eigenvoice matrix and diagonal residual, respectively), and $\mathbf{U}$ defines a session subspace (eigenchannel matrix). The vectors $\mathbf{y}$, $\mathbf{z}$ and $\mathbf{x}$ are the speaker- and session-dependent factors in their respective subspaces and each is assumed a random variable with a Normal distribution $\mathcal{N}$. The application of JFA to speaker recognition consists of firstly estimating the subspaces (i.e., $\mathbf{V}$, $\mathbf{D}$, $\mathbf{U}$) from appropriately labeled training corpora and then estimating the speaker and session factors (i.e., $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$) for a given new target utterance. The speaker-dependent supervector is given by

$$\mathbf{s} = \mathbf{m} + \mathbf{V}\mathbf{y} + \mathbf{D}\mathbf{z}. \tag{2.7}$$

Scoring is done by computing the likelihood of the test utterance feature vectors against a session-compensated speaker model ($\mathbf{M} - \mathbf{U}x$) [5].

2.3.2 iVector

iVector system is a generalized form of the JFA. We assume that speaker- and session-dependent GMM supervector $\mathbf{s}$ can be modeled as:

$$\mathbf{s} = \mathbf{m} + \mathbf{T}\mathbf{w} \tag{2.8}$$

where $\mathbf{m}$ is the UBM GMM mean supervector, $\mathbf{T}$ is a low-rank matrix representing M bases spanning subspace with important variability in the mean supervector space, and $\mathbf{w}$ is a standard normal distributed vector of size M.

We can see that method described in the 2.8 does not have two separate factors for speaker and channel variability. Instead of that, it has only one factor *total variability* describing speaker and channel variability together. For each observation $\mathcal{X}$, the aim is to estimate the parameters of the posterior probability distribution of variable $\mathbf{w}$[1]:

$$\mathrm{p}(\mathbf{w}|\mathcal{X}) = \mathcal{N}(\mathbf{w}; \mathbf{w}_\mathcal{X}, \mathbf{L}_\mathcal{X}^{-1}), \tag{2.9}$$

where $\mathbf{w}_\mathcal{X}$ is a mean of the variable $\mathbf{w}$ and $\mathbf{L}_\mathcal{X}^{-1}$ is a covariation matrix of posterior probabilities of the variable $\mathbf{w}_\mathcal{X}$. The iVector is the MAP point estimate of the variable $\mathbf{w}$, i.e. the mean $\mathbf{w}_\mathcal{X}$ of the posterior distribution $\mathrm{p}(\mathbf{w}|\mathcal{X})$. It maps the most of the relevant information from a variable-length observation $\mathcal{X}$ to a fixed- (small-) dimensional vector. $\mathbf{T}$ is referred to as the iVector extractor.

The input data for the observation $\mathcal{X}$ is given as a set of *zero-* and *first-order statistics* — $N_\mathcal{X}$ and $f_\mathcal{X}$. For each Gaussian component c, the statistics are given respectively as:

$$N_\mathcal{X}^{(c)} = \sum_t \gamma_t^{(c)}, \tag{2.10}$$

$$\mathbf{f}_\mathcal{X}^{(c)} = \sum_t \gamma_t^{(c)} \mathbf{x}_t, \tag{2.11}$$

[1] This w is not the w_c as Gaussian weight

where $\mathbf{x}_t$ is the feature vector in time t, and $\gamma_t^{(c)}$ is its occupation probability from Equation 2.4. The complete zero- and first-order statistics supervectors are $\mathbf{f}_\mathcal{X} = \left(\mathbf{f}_\mathcal{X}^{(1)\prime}, \ldots, \mathbf{f}_\mathcal{X}^{(C)\prime}\right)'$, and $n_\mathcal{X} = \left(N_\mathcal{X}^{(1)}, \ldots, N_\mathcal{X}^{(C)}\right)'$.

As described in [3] and [7], for an observation $\mathcal{X}$, the corresponding iVector is computed as a point estimate:

$$\mathbf{w}_\mathcal{X} = \mathbf{L}_\mathcal{X}^{-1}\mathbf{T}'\mathbf{f}_\mathcal{X} \tag{2.12}$$

where $\mathbf{L}$ is the precision matrix of the posterior distribution [7], computed as:

$$\mathbf{L}_\mathcal{X} = \mathbf{I} + \sum_{c=1}^{C} N_\mathcal{X}^{(c)}\mathbf{T}^{(c)\prime}\mathbf{T}^{(c)} \tag{2.13}$$

.

2.3.3 PLDA

iVectors $\mathbf{w}_\mathcal{X}$ are assumed to be distributed according to the form

$$\mathbf{w}_\mathcal{X} = \mathbf{m} + \mathbf{V}\mathbf{y} + \mathbf{U}\mathbf{x} + \epsilon, \tag{2.14}$$

incorporating speaker $\mathbf{V}$ and channel $\mathbf{U}$ subspaces [9].

To facilitate comparison of iVectors in a verification trial, we model the distribution of i-vectors using a Probabilistic LDA model [10, 4]. We first consider only a special form of PLDA, a *two-covariance model*, in which speaker and inter-session variability are modeled using across-class and within-class full covariance matrices $\mathbf{\Sigma}_{ac}$ and $\mathbf{\Sigma}_{wc}$. The two-covariance model is a generative linear-Gaussian model, where latent vectors $\mathbf{y}$ representing speakers (or more generally classes) are assumed to be distributed according to prior distribution

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mu, \mathbf{\Sigma}_{ac}). \tag{2.15}$$

For a given speaker represented by a vector $\hat{\mathbf{y}}$, the distribution of iVectors is assumed to be

$$p(\phi|\hat{\mathbf{y}}) = \mathcal{N}(\phi; \hat{\mathbf{y}}, \mathbf{\Sigma}_{wc}). \tag{2.16}$$

The ML estimates of the model parameters, μ, $\mathbf{\Sigma_{ac}}$, and $\mathbf{\Sigma_{wc}}$, can be obtained using an EM algorithm as in [4]. The training iVectors come from a database comprising recordings of many speakers (to capture across-class variability), each recorded in several sessions (to capture within-class variability).

In the more general case, the speaker and/or inter-session variability can be modeled using subspaces [2] as in Equation 2.14. For example, in our baseline system, speaker variability is not modeled using a full covariance matrix. Instead, a low rank across-class covariance matrix is modeled as $\mathbf{\Sigma_{ac}} = \mathbf{V}^T\mathbf{V}$, which limits speaker variability to live in a subspace spanned by the columns of the reduced rank matrix $\mathbf{V}$.

Evaluation of Verification Score

Consider the process of generating two iVectors ϕ_1 and ϕ_2 forming a trial. In the case of a same-speaker trial, a single vector $\hat{\mathbf{y}}$ representing a speaker is generated from the prior $p(\mathbf{y})$, for which both ϕ_1 and ϕ_2 are generated from $p(\phi|\hat{\mathbf{y}})$. For a different-speaker trial, two vectors representing two different speakers are independently generated from $p(\mathbf{y})$. For each,

one of the iVectors ϕ_1 and ϕ_2 is generated. Given a trial, we want to test two hypotheses: $\mathcal{H}_d$ that the trial is a different-speaker trial and $\mathcal{H}_s$ that the trial is a same-speaker trial. The speaker verification score can now be calculated as a log-likelihood ratio between the two hypotheses $\mathcal{H}_s$ and $\mathcal{H}_d$ as

$$s = \log \frac{p(\phi_1, \phi_2|\mathcal{H}_s)}{p(\phi_1, \phi_2|\mathcal{H}_d)} \tag{2.17}$$

$$= \log \frac{\int p(\phi_1|\mathbf{y})p(\phi_2|\mathbf{y})p(\mathbf{y})d\mathbf{y}}{p(\phi_1)p(\phi_2)}, \tag{2.18}$$

where in the numerator, we integrate over the distribution of speaker vectors and, for each possible speaker, the likelihood of producing both iVectors from the speaker is calculated. In the denominator, we simply multiply the marginal likelihoods $p(\phi) = \int p(\phi|\mathbf{y})p(\mathbf{y})d\mathbf{y}$. The integrals, which can be interpreted as convolutions of Gaussians, can be evaluated analytically giving

$$\begin{aligned} s &= \log \mathcal{N}\left(\begin{bmatrix}\phi_1\\ \phi_2\end{bmatrix}; \begin{bmatrix}\mu\\ \mu\end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma_{tot}} & \mathbf{\Sigma_{ac}} \\ \mathbf{\Sigma_{ac}} & \mathbf{\Sigma_{tot}} \end{bmatrix}\right) \\ &- \log \mathcal{N}\left(\begin{bmatrix}\phi_1\\ \phi_2\end{bmatrix}; \begin{bmatrix}\mu\\ \mu\end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma_{tot}} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma_{tot}} \end{bmatrix}\right), \end{aligned} \tag{2.19}$$

where the total covariance matrix is given as $\mathbf{\Sigma_{tot}} = \mathbf{\Sigma_{ac}} + \mathbf{\Sigma_{wc}}$. By expanding the log of Gaussian distributions and simplifying the final expression, we obtain

$$\begin{aligned} s &= \phi_1^T \Lambda \phi_2 + \phi_2^T \Lambda \phi_1 + \phi_1^T \Gamma \phi_1 + \phi_2^T \Gamma \phi_2 \\ &+ (\phi_1 + \phi_2)^T \mathbf{c} + k, \end{aligned} \tag{2.20}$$

where

$$\begin{aligned} \Gamma &= -\frac{1}{4}(\mathbf{\Sigma}_{wc} + 2\mathbf{\Sigma}_{ac})^{-1} - \frac{1}{4}\mathbf{\Sigma}_{wc}^{-1} + \frac{1}{2}\mathbf{\Sigma}_{tot}^{-1} \\ \Lambda &= -\frac{1}{4}(\mathbf{\Sigma}_{wc} + 2\mathbf{\Sigma}_{ac})^{-1} + \frac{1}{4}\mathbf{\Sigma}_{wc}^{-1} \\ \mathbf{c} &= ((\mathbf{\Sigma}_{wc} + 2\mathbf{\Sigma}_{ac})^{-1} - \mathbf{\Sigma}_{tot}^{-1})\mu \\ k &= \log|\mathbf{\Sigma}_{tot}| - \frac{1}{2}\log|\mathbf{\Sigma}_{wc} + 2\mathbf{\Sigma}_{ac}| - \frac{1}{2}\log|\mathbf{\Sigma}_{wc}| \\ &+ \mu^T(\mathbf{\Sigma}_{tot}^{-1} - (\mathbf{\Sigma}_{wc} + 2\mathbf{\Sigma}_{ac})^{-1})\mu. \end{aligned} \tag{2.21}$$

We can precompute values $\Gamma, \Lambda, \mathbf{c}$ and k from the Equation 2.21 on the training data and use them as constants.

2.3.4 Mathematics and Code

- *Preprocessing* First, we extract parts of the input sound that belongs to the speech. This decision is made by VAD 2.2.1 which is implemented in these set of modules 5.1.1. These parts are joined together and sent to the next stage - Feature extraction subsystem.
- *Feature extraction* Input of the iVector system - MFCC+Deltas+STG are obtained in the standard way. Melbanks 5.1.2 are created from the waveform after preprocessing by VAD. From Melbanks MFCCs are created 5.1.2. Derivations (deltas and double deltas) 5.1.2 are approximated from the MFCCs and after that Short Time Gaussianization 5.1.2 is performed.

- *Statistics* From the input features, we create $\gamma_t^{(c)}$ in (2.4) and then we extract zero- (2.10) and first-order (2.11) statistics and store them in the buffer 5.1.3 for iVector extractor.

- *iVector Extraction* Statistics are passed to the iVector extractor in fixed intervals (or on demand). Output of the iVector Extractor module 5.1.3 is an iVector (2.13 , 2.12) with fixed length. We can simple store it as a model of speaker in the enrollment mode. In the test mode, we pass it to the PLDA scoring part.

- *Scoring* We want to compare two iVectors in the testing mode to verify or reject speaker against pre-enrolled model (iVector trained in the previous stage). We are using modification of PLDA implemented in module 5.1.3 for this task. Our optimized 6 PLDA implementation (in the form of Two-Covariance Model) uses precomputed constant matrixes (2.21) estimated on training data to generate score (2.20). Output from this last stage is a soft-decision - Log-Likelihood Ratio.

Chapter 3

MOBIO Project and Target Platform

MOBIO is a consortium of universities, research centers and companies joined in the EU-funded project MOBIO [1]. The goal of this project is to bring robust biometric identification to the mobile devices with the common equipment (microphone and camera).

3.1 Project Description

Project is focused to the following particular targets.

- Advanced research and development on joint bi-modal authentication, involving development of new statistical models actually processing both channels simultaneously and in a principled way.
- Investigation of model adaptation techniques to reduce the degradation of biometric systems over time
- Analyzing the scalability of the proposed solutions by studying how the performance of the system degrades while the complexity of the model is reduced.
- Providing common evaluation tools and baseline results to the research community in order to evaluate and compare the developed technologies.

The project has two main scenarios, where the developed identification systems can be used:

- Embedded biometry where the Bi-Modal Biometric Authentication - BMBA system is running entirely on a mobile phone. The system is designed to maximize the authentication performance and to minimize resources such as CPU, memory and speed.
- Remote biometry if the BMBA system needs too many resources to reach the required performance it will be hosted on a server while a minimum of essential functionalities would stay on the mobile phone such as capture, segmentation, preprocessing and feature extraction.

[1] http://www.mobioproject.org

3.2 Partners

Each partner in the consortium has specified own part of work to develop. Overview of how the project goals were allocated is summarized in table 3.1.

Partner	Function in the project
Idiap Research Institute (CH)	Coordinator, Face localization and verification
University of Manchester (UK)	Face Localization
University of Surrey (UK)	Fusion of FV and SV scores
University of Avignon (FR)	Speaker verification
Brno University of Technology (CZ)	Speaker verification
University of Oulu (FN)	Face detection and verification
IdeArk (CH)	Dissemination and organization of community of interest
Visidon (FI)	Implementation of the mobile framework

Table 3.1: Summarization of the MOBIO partners

3.3 Hardware

As this work is targeted on the mobile phone, it was important to choose proper mobile device where our application is able to run. Because both systems (speech and face) have high requirements in the terms of performance, the consortium chooses Nokia N900 as the target device. It was in fact the only device on the market capable to run our algorithms.

3.3.1 Nokia N900

Nokia N900 [2] is (as in 2010), the most powerful device from Nokia. It has many capabilities unusual for the mobile phones. Especially, it has common Linux OS - Maemo 5, which is modified branch of Debian. It is possible to use classic GStreamer interface for accessing audio and video devices. The code is also highly portable from the PC - we can use the same code; changing of the compiler makes it suitable for Nokia N900.

Specifications

Key specifications, important for the project, are summarized in Table 3.2.

Nokia N900 has also many other features, which are unimportant for our task, but still interesting, e.g.: Wi-Fi, A-GPS, Bluetooth, 1800/1900/850/900 MHz GSM and 900/1700/2100 MHz WCDMA, Touchscreen, Main 5MPx Camera, Graphics with OpenGL ES 2.0 support etc.

3.3.2 Development Tools

Nokia provides complete development tools set - Maemo SDK, which can be installed from Maemo web-site[4]. It is targeted to run on Debian-like system, include Debian itself, but use of Ubuntu Linux is recommended. It consists of the following parts:

[2] http://maemo.nokia.com/n900/

[4] http://maemo.org/development/sdks/

CPU Type	ARM Cortex-A8
FPU	Neon capable
Processor, TI OMAP	3430
CPU Clock Rate	600 MHz
NAND Memory	768 MB
SDRAM Memory	256 MB
Frontal Camera Resolution	640 x 480 px
Screen Resolution	800 x 480 px
Keypad	Slide-out QWERTY Keyboard

Table 3.2: Specifications of Nokia N900 [3]

- Xephyr - X server for emulator
- Scratchbox - cross-compiler system for many platforms
- Nokia binaries - closed-source platform-dependent Nokia libraries and applications

Scratchbox

Scratchbox is a cross-compilation toolkit designed to make embedded Linux application development easier. It also provides a full set of tools to integrate and cross-compile an entire Linux distribution[5].

Scratchbox physically creates jail root for each maintained platform target. It means that the application, which runs inside the jail, cannot observe, that it is not on the physical root file system - it is a kind of virtualization. This solution enables maintaining unlimited number of virtual targets (only one limitation is the free space on disk). In this project, we used two targets: x86 and ARM.

3.3.3 Limitations

Nokia N900 is a mobile device based on ARM architecture with limited resources, so there are a few limitations that we have to consider:

Memory

As stated in Table 3.2, there is 256 MB SDRAM and 768 MB NAND memory. Together, we have 1024 MB RAM memory. Unfortunately, only SDRAM memory is pure RAM, the other 768 MB NAND is swapped on the internal SD card, thus it has much longer access times than SDRAM. We have to manage to fit our application into the first 256 MB of memory, if we want maximum of performance. It is usually a problem, because when Maemo 5 system boots, it takes about 160 MB of RAM, therefore only 90 MB is available for the application usage.

CPU

ARM processors are not generally weaker than x86 CPUs, but there is a small complication with the compiler. GCC, which is default Linux compiler for a various platforms, has a

[5] http://www.scratchbox.org/

problem with FPU (Floating Point Unit) utilization, although FPU is present on the device. Thus FPU usability is implicitly very poor and it is necessary to make changes in the source code and replace FPU operations with NEON intrinsics (inline assembler commands for FPU also called Advanced SIMD) to use its full power (we will discuss this later in Section 6.5).

Chapter 4

Software Environment

4.1 Implementation Language

The MOBIO consortium decided to use C++ as the main implementation language. It was a logical choice - native language of Linux is C/C++, which is also native language of Nokia N900. Every member of consortium also had its own development tool in C (or C++) and finally C++ has many advantages in terms of speed and object-oriented programming.

4.2 BS-CORE

Brno Speech Core developed jointly by Phonexia[1] and BUT is a set of building stones for easy and fast prototyping of speech recognizers. It implements wide scale of algorithms from reading of speech files/microphone input, processing of list files through parameterization, feature transformation, classification, decoding, phoneme recognizers to continue speech recognizers, keyword spotting, language identification, speaker identification. Brno Speech Application Interface (BS-API) is an interface between BS-CORE and other software.

In BS-API, there are many function classes/modules, which are possible to connect to each other with various types of callbacks. BS-API architecture was inspired by COM (Common Object Model).

4.3 MOBIO Framework

The MOBIO partner Visidon defined the general framework. Figure 4.1 illustrates the basic architecture of MOBIO A/V framework. The sources (camera, microphone, video file) are opened and processed via GStreamer. This procedure is used for obtaining raw data samples that are then fed to the separate voice and facial verification modules. The biometric modules itself are implemented with a standard C++ interface, providing easy integration.

Figure 4.2 illustrates the basic architecture of speaker verification module. This module provides two main functionalities: enrollment and verification. In the enrollment mode, SV library creates the speaker model from the input audio samples. Enrollment is run until enough data is processed by module. In the verification mode, the module is firstly used to load the enrolled speaker model, and then iteratively called to verify speaker against the

[1] http://www.phonexia.com/

loaded model. API for the Speaker Verification module is defined in SpeakerVerification.h (See Appendix A).

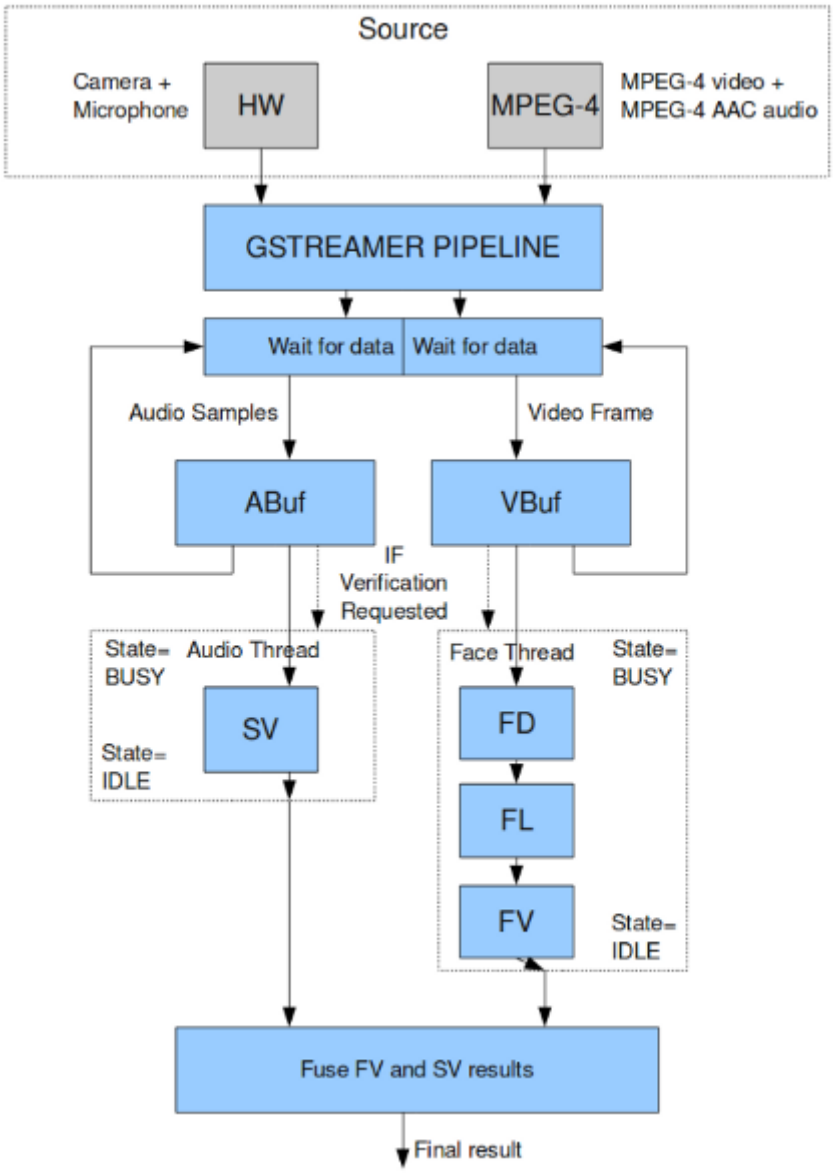


Figure 4.1: Structure of MOBIO Framework as Defined in Deliverable D6.2 [15]

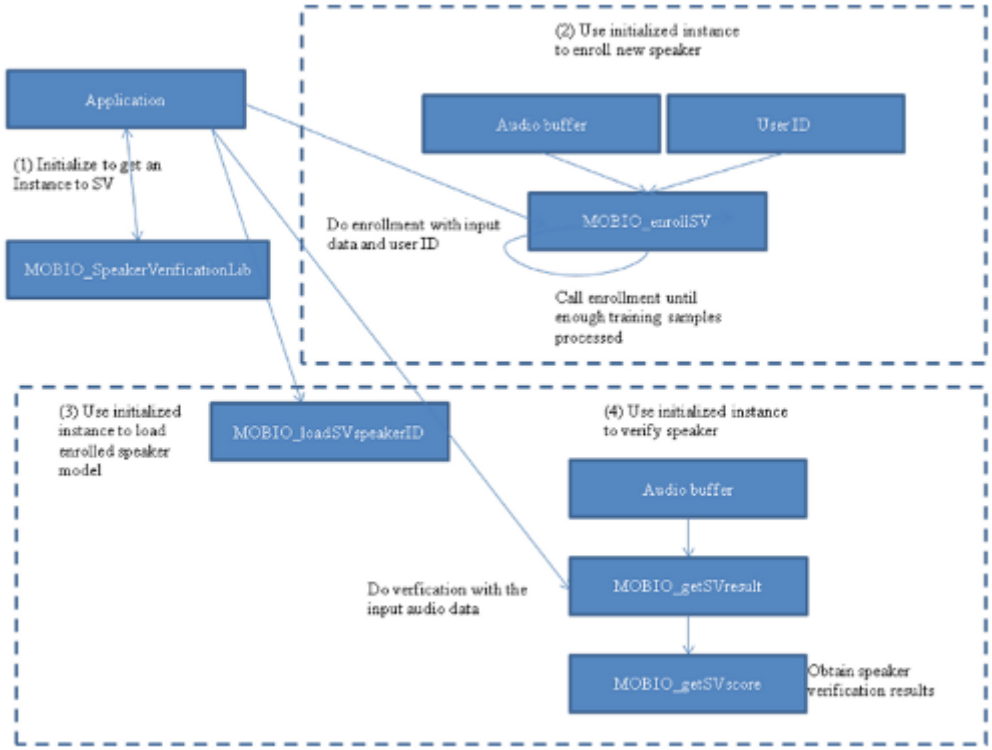


Figure 4.2: Structure of MOBIO Framework API for Speaker Verification Module As Defined in Deliverable D6.2 [15]

Chapter 5

Implementation

The whole SID system was based on BS-API baseline SID system 2.2. The final parts of the processing line were reimplemented to use iVector system. Scheme of the complete system is available Figure 5.2. Implementation was done during the D6.4 [13] stage of MOBIO project (11/2009 - 11/2010).

5.1 System Structure

System scheme is described using two figures. Figure 5.1 represents symbolic scheme of structure. In figure 5.2 physical module implementation with the BS-CORE modules is shown.

5.1.1 Preprocessing

SFileWaveformSourceI

This module is the entry point of the whole processing. It is normally used in the off-line mode to process file inputs. In this project, I modified it in order to take the input and send it piecewise, as it comes from the microphone input. It was used as debug and testing utility. In the final system in framework, this module is inactive.

SWaveFormFormatConvertorI

This module is the entry point of the whole processing. It receives wave stream consisting of 2 x short integer (each 8bit) per frame[1] (linear 16-bit, 8 kHz) and converts them to the stream of floats.

SWaveFormSplitterI

This module acts as the simple splitter that sends data to the two consequent modules: VAD input and SWaveFormOnlineSegmenterI.

VAD Subsystem

It consists of the following modules:

[1] This means audio raw frame, not the speech frame with overlapping!

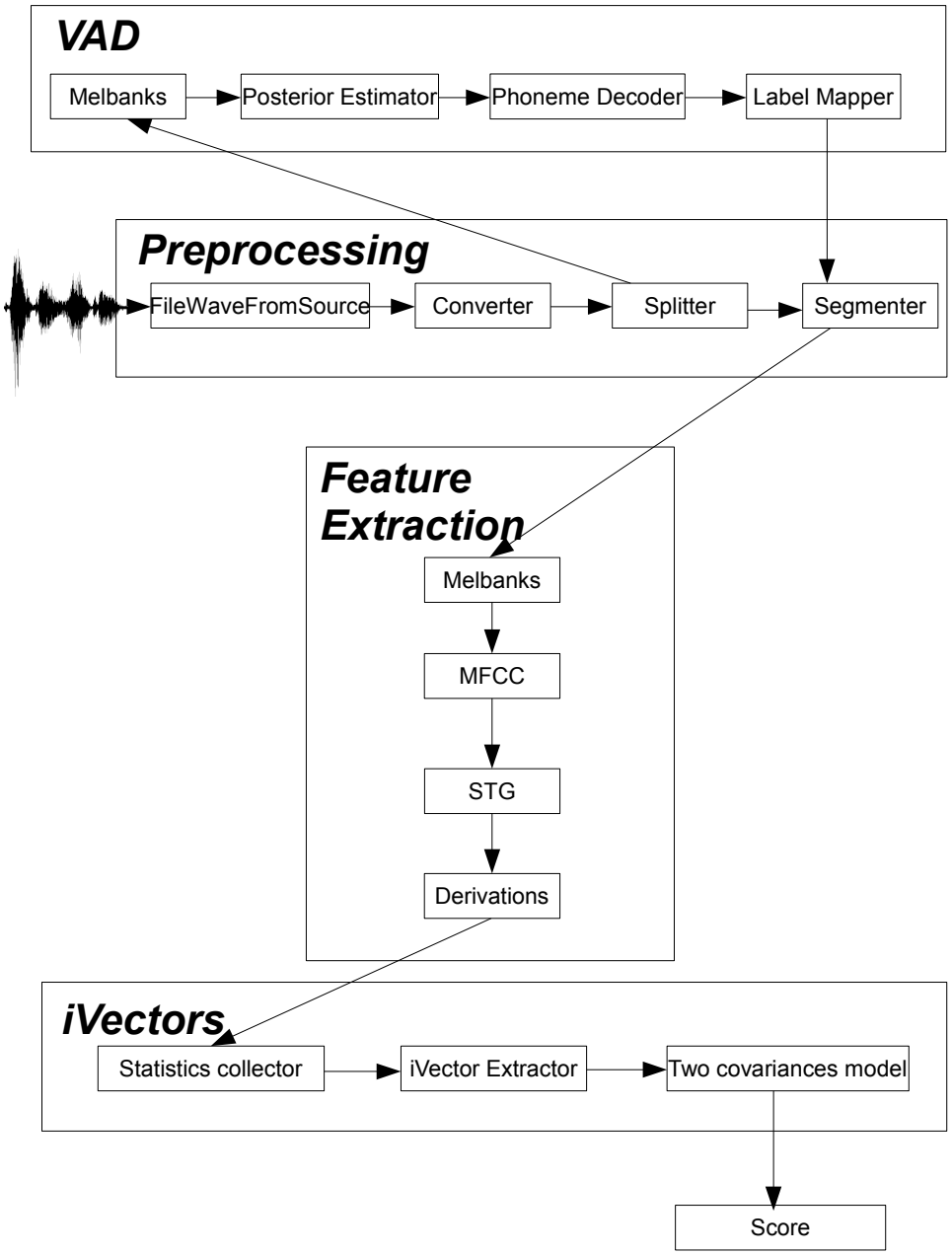


Figure 5.1: Symbolic Structure of SID iVector Mobile System

1. SMelBanksI
2. SNNetPosteriorEstimatorI
3. SPhnDecoderI
4. SLabelMapperI

Neural network (NN) inside the VAD uses split temporal context as in [14]. It has 2 „context“ neural nets for the each split context and on the output of this nets is last net called „merger“. Each net has three layers: input, hidden and output. Context nets have 165 input neurons, 50 hidden neurons and 33 output neurons. Merger has 66 input neurons, 50 hidden neurons and 33 output neurons. Output neurons of the merger are linked to the 33 phoneme classes from the Hungarian phoneme set (including silence and pauses). Viterbi decoder is used to choose the best phoneme sequence. It has configurable length of history and it is connected to the output of the merger net. All phoneme classes returned by Viterbi decoder are linked to *speech* class and the others are linked to the *silence* class. Segmentation is then forwarded further.

SWaveFormOnlineSegmenterI

The original Segmenter block was designed only for the offline mode, so I had to rewrite it to the online mode. Segmenter has to wait in online mode, because the VAD block has non-zero delay due to decoder sub-block. Waiting for the segmentation is implemented as a circular buffer with known length (it can be defined from the processing scheme of VAD). Audio frames are stored in this buffer until corresponding segmentation arrives from the VAD.

Segmenter receives two data streams:

- Segmentation data stream from the VAD.
- Wave data stream from the Splitter.

Wave stream frames are stored to keep both modules synchronized (wave stream has no delay). Actual setting of the VAD delay is 50 frames history (it gives 500ms delay to real-time). When Segmenter receives *speech* segmentation it forwards corresponding frame to the next module, on the other hand it discards each *silence* frame.

5.1.2 Feature Extraction

SMelBanksI

To this module comes only speech frames because of VAD and Segmenter. It computes 256-point FFT and applies 25 mel-banks from 300 to 3400 Hz.

SMFCCI

This module creates 19 Mel-frequency cepstral coefficients from the input frames passed by SMelBanksI module.

SSTGI

Short Time Gaussianization is conducted here with 50 frames left and 5 frames right context. Context is a result of our experiments targeted on minimization of the delay of the whole system. Final delay to the real-time in this block is 50ms (length of the shifting window).

SFeatureDerivationsI

Derivations module computes the first and the second derivations of the MFCC trajectories coming to it. This module generates another 4 frames (40ms) delay.

5.1.3 Statistical modeling

SGMMStatisticCollectorI

Zero- and first-order statistics from the MFCC coefficient trajectories using UBM are computed here. UBM was trained on NIST 2004 and NIST 2005 evaluation data. Zero order statistics are computed with Equation 2.10 and the first order statistics are computed with Equation 2.11.

This block is capable to send statistics as one compact matrix forward. I had to implement an asynchronous trigger, which can send statistics matrix on demand. It was necessary because of the time and resources consuming by the next block - iVector Extractor.

SIvectorExtractorI

This module takes statistics from the *Statistics collector* module and extracts iVector of size 400 from them 2.3.4. Extractor was trained on NIST 2004, NIST 2005, NIST 2006 evaluation data and on Fisher English 1,2.

STwoCovarianceModelI

The iVector is scored against speaker model iVector using PLDA in this module. Speaker model iVector is pre-trained in the enrollment mode and loaded from the storage. Output of the PLDA is a Log-Likelihood Ratio score of the iVector passed to the framework, where it is fused with face score. Model was trained on NIST 2004, NIST 2005 and NIST 2006 evaluation data.

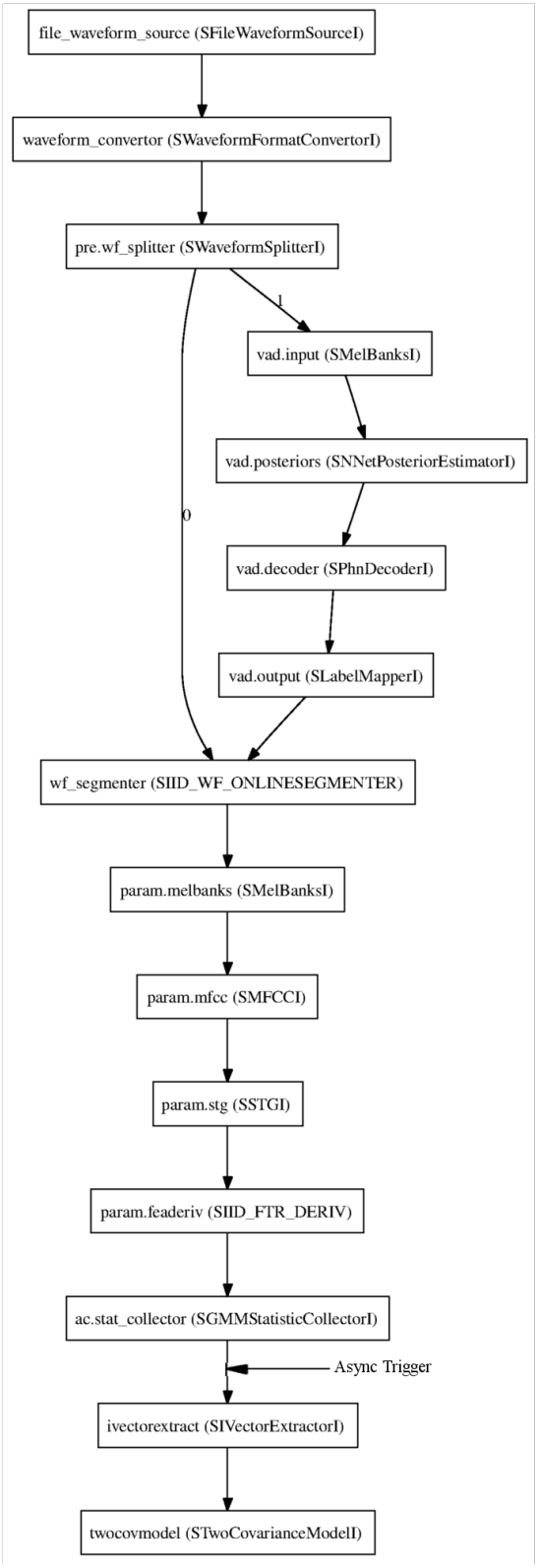


Figure 5.2: Scheme of Physical Module Structure of the System

Chapter 6

Performance and Optimization

6.1 System Parameters

The system that we implemented into mobile phone as initial try is described in Table 6.1. System time consumption on the N900 is available Table 6.2 It was necessary to tune and optimize this system up, in order to make it capable to run in the mobile environment in real-time. In this chapter, I will describe how we manage to do performance optimization.

Number of features	60
Number of Gaussians	512
Size of supervector	30720
Size of iVector	400
Total delay to the real-time	590 ms

Table 6.1: Initial System Implemented to the Mobile Phone.

Processing part	Time (s)
System initialization	140
Feature extraction	55
iVector extraction	315
TwoCovariance Scoring (PLDA)	90
TOTAL	600

Table 6.2: Profiling of Initial System.

6.2 Matrix Operations

First of all, I had to rewrite basic matrix arithmetic manipulation procedures from the ATLAS (Automatically Tuned Linear Algebra Software) [1] on which BS-CORE relies, to the standard C. Unfortunately this modification significantly slowed down matrix operations, but it was the only way to implement it, because ATLAS was not available for the ARM platform.

[1] http://math-atlas.sourceforge.net/

It was necessary to rewrite main matrix operation GEMM (General Matrix Multiply). It calculates the new value of matrix **C** based on the matrix-product of matrices **A** and **B**, and the old value of matrix **C**[2].

$$C \leftarrow \alpha \mathbf{A} B + \beta \mathbf{C} \tag{6.1}$$

After the first implementation of the whole iVector system and successful run on the mobile phone, many performance issues emerged. Processing of a 28s recording took 600s on the device, which means it was 21.5x slower than real-time.

6.3 Debugging the Original System

Because BLAS library is not present on the ARM architecture, we get about 15x slower performance on matrix operations (in floating point). I had to enable suitable parameters for utilizing internal FPU coprocessor:

```
-mfloat-abi=softfp -ftree-vectorize -ffast-math
```

After enabling these parameters for the compiler, performance gap falls down to the 10x slower. Even if it was better, it was still impossible to reach real-time level, so we have to make deeper optimizations inside the system.

6.4 Algorithmic Optimization

We reveal that Feature Extraction in the performance tests consumes only 5% of the time. Remaining 95% was consumed in the iVector subsystem. We can see that the computational complexity of the whole estimation for one observation is $O(CFM + CM^2 + M^3)$, where C represents number of Gaussians, F represents number of input data and M stand for the size of iVector. The first term represents the $\mathbf{T}'\mathbf{f}_{\mathcal{X}}$ multiplication in (2.12). The second term represents the sum in (2.13) and includes the multiplication of $\mathbf{L}_{\mathcal{X}}^{-1}$ with a vector. The third term represents the matrix inversion.

The memory complexity of the estimation is $O(CFM + CM^2)$. The first term represents the storage of all the input variables in (2.12), and the second term represents the precomputed matrices in the sum of (2.13).

Note that the computation complexity grows quadratically with M in the sum of (2.13), and linearly with C. This is definitely a bottleneck of our system.

To limit the complexity, we apply the assumption that the GMM component alignment is constant across segments, i.e. the posterior occupation probabilities $\gamma^{(c)}$ in (2.10) are replaced by their prior probabilities represented by the UBM GMM weights. The new zero-order statistics are then:

$$\bar{N}_{\mathcal{X}}^{(c)} = \omega^{(c)} N_{\mathcal{X}} \tag{6.2}$$

where $\omega^{(c)}$ is the GMM UBM weight of component c, and $N_{\mathcal{X}} = \sum_{j=1}^{C} N_{\mathcal{X}}^{(j)}$. Substituting $N_{\mathcal{X}}^{(c)}$ in (2.13) by $\bar{N}_{\mathcal{X}}^{(c)}$ from (6.2), we get

$$\bar{\mathbf{L}}_{\mathcal{X}} = \mathbf{I} + N_{\mathcal{X}} \mathbf{W} \tag{6.3}$$

[2] http://en.wikipedia.org/wiki/General_Matrix_Multiply

where

$$\mathbf{W} = \sum_{c=1}^{C} \omega^{(c)} \mathbf{T}^{(c)\prime} \mathbf{T}^{(c)} \tag{6.4}$$

Note that $\mathbf{W}$ in (6.4) is independent of data and thus it was precomputed and stored in SD card memory. Its resulting size is $M \times M$, yielding faster computation and less memory demands. The computational complexity of this algorithm reduces to $O(CFM + M^3)$ with the dominating inversion step. The memory complexity reduces to $O(CFM + M^2)$.

6.5 Code Optimization

After the algorithmic optimizations were done, our system still was not able to run smoothly. We decided to rewrite the most CPU consuming part, dot product, to the NEON FPU intrinsics. NEON coprocessor is enabled by the following compiler parameters:

```
-mfpu=neon -mcpu=cortex-a8
```

Sample code that does matrix dot product in NEON intrinsics is in listing 6.1.

```
1   float32x2_t tmp3;
2   float32x2_t p1, p2;
3
4   tmp3 = vdup_n_f32(0.0);
5
6   for(int j=0;j<nC;j+=2)
7   {
8     p1 = vld1_f32(reinterpret_cast<const float32_t*>(pmV));
9     p2 = vld1_f32(reinterpret_cast<const float32_t*>(pmF));
10    tmp3 = vmla_f32(tmp3,p1,p2);
11    pmF += 2;
12    pmV += 2;
13  }
14  tmp3 = vpadd_f32(tmp3,tmp3);
15  tmp = vget_lane_f32(tmp3,0);
```

Listing 6.1: Example of code implemented in NEON FPU intrinsics

Even though we made many optimizations in our system it was still too slow to run in the mobile phone in the real-time, so we had to downscale our system for this application. In table [6.4] we can see that 128G (system with 128 Gaussians) with algorithmic optimizations and precomputed matrix system gives us reasonable results in terms of speed for the processing on the test recording.

6.6 Data Optimization

TwoCovariance Model part of the system was also too slow (based on results in Table 6.2). We can see in the Eq. 2.21 that the constants $\Gamma, \Lambda, \mathbf{c}$ and k are independent of the data, so we precomputed it into internal memory. This gives us little bit faster startup of the system and faster processing of PLDA. Profiling of the final system is in Table 6.3.

6.7 Other Optimization

When we connect our SID system to the MOBIO framework, it was not once again incapable of running smoothly due to other subsystems running in the framework (Face localization, Face detection, Face verification and Framework itself). We have only less than a half of the CPU power available and extraction of the iVector was still the most consuming part of the processing scheme. After that, we managed to split the processing scheme and move the extraction part to an asynchronous thread the final structure of the implemented system is in Figure 6.1.

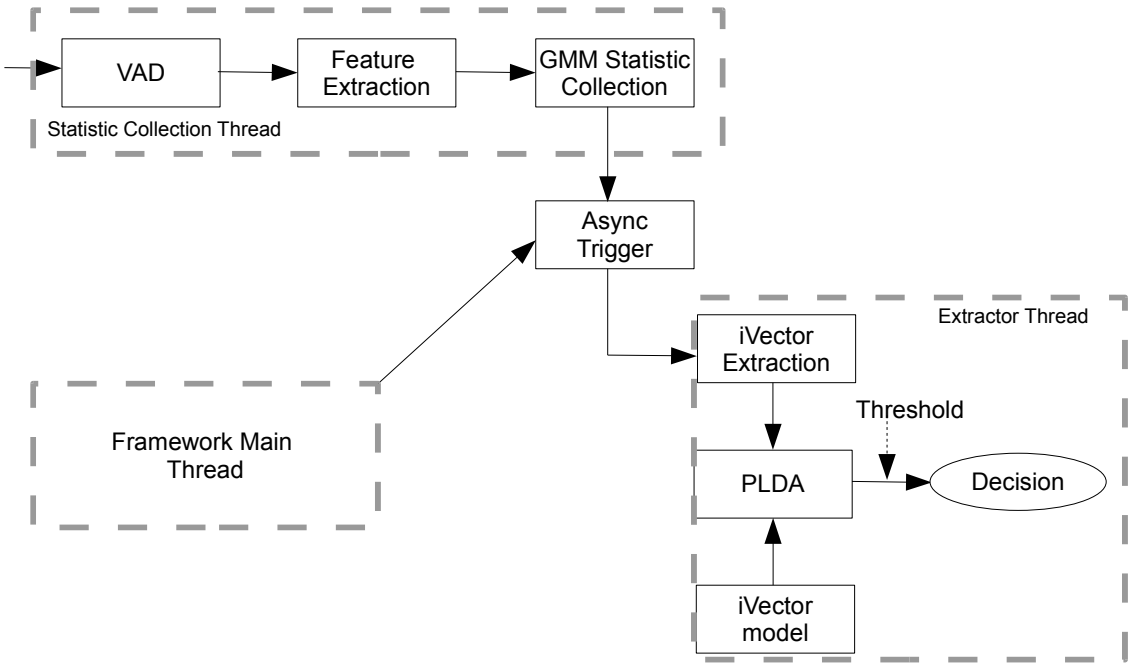


Figure 6.1: Final Implemented and Optimized Structure of the SID System

We modified the framework to send signals to the extraction thread only in fixed intervals (to give a time to extractor for successful extraction of iVector). The main thread in the background constantly c ollects the statistics and it passes statistics to the extractor on the control command from the framework thread. Extractor slave thread computes iVector and makes a decision.

Sometimes the load on CPU was still too high. Therefore, we managed to successfully over-clock the stock Nokia N900 CPU from 600 MHz to 1 GHz. Finally, this gives us enough power to get our system to work in the real-time mode.

Processing part	Time (s)
System initialization	2
Feature extraction	11
iVector extraction	1
TwoCovariance Scoring (PLDA)	0.5
TOTAL	14.5

Table 6.3: Profiling of the Final System.

	Memory (MB)	emulator (s)	**N900 (s)**	PC (s)
512G	47	X	**600**	3
256G	23	X	**312**	1
128G	11	X	**160**	<1
512G_p	359	20	**24**	2
256G_p	180	16	**13**	<1
128G_p	90	8	**7**	<1
512G_pa	48	9	**10**	1
256G_pa	24	9	**8**	<1
128G_pa	12	6	**5**	<1

Table 6.4: Downscaling and Optimizing Results (Memory consumption and run-times on N900 emulator, real N900 and standard PC. (p=precomputed, pa=precomputed and algorithmic optimized)

Chapter 7

Tests and Results

We used two data sets from different sources for measuring performance of our modified SID.

7.1 Data Sets

7.1.1 NIST

NIST data set is a data set from the NIST SRE 2010 competition [6]. In this particular test we used data from telephone set condition 5 extended. Length of the speech of each person in one recording is approximately 2.5 minutes (5 minutes long conversation). We used only female part of the set. It has 3704 target and 233077 nontarget trials.

7.1.2 MOBIO

This bi-modal database was captured in two phases and consists of 152 participants with a female to male ratio of nearly 1:2 (100 males and 52 females). Each session recorded for Phase I consists of 21 questions which the user was prompted to answer. These questions varied from set responses, read speech from a paper through to, to free speech. It contains 1650 target and 34650 nontarget female trials. On male part, it contains 2925 target and 111150 nontarget trials.

During the second phase (Phase II), six more sessions were recorded from 152 participants[1]. Each session for Phase II consists of 11 questions and includes the same variation as the one captured in Phase I. [8] It contains 196 target and 3332 nontarget female trials. On male part, it contains 130 target and 4810 nontarget trials.

Protocol

The protocol divides speaker samples into 3 parts.

- training set - derivation of background models (We used NIST data to train UBM.)
- development set - derivation of fusion parameters (We used this set only for training speaker models, fusion parameters were used in the fusion part of the project)
- test - verification of results

[1] Compared to the first phase, eight participants were not able to take part in the second phase of the recording

MOBIO data were recorded in normal conditions with slight noise on the background. In addition, length of the utterances is a problem, because average length is about 5 seconds, which is too short for the normal system. Especially for training of model, there is a need of utterances at least 30 seconds long. Therefore, we had to join each utterance from one speaker together for training the model. Testing was done on the unmodified utterances.

7.2 Results

7.2.1 NIST

We made tests on this set using similar system as in mobile phone (iVector system). The only difference was that it ran in the offline mode. Tests were done for 3 sizes of the system (in terms of number of Gaussians) and measured with Equal Error Rate metrics. To compare with fully blown *state-of-the-art* systems, we completed the table with the best performing systems from the NIST 2010 evaluation used by BUT[1]. NIST 2048G is 4x bigger and uses normalizations. ABC NIST fusion is a fused system also from the NIST 2010, result of this system is composed from various systems, joined together.

System	EER (%)
MOBIO 128G	7.06
MOBIO 256G	6.89
MOBIO 512G	6.74
NIST 2048G	2.06
ABC NIST fusion	2.33

Table 7.1: Results on NIST Data Set

7.2.2 MOBIO

On MOBIO dataset, we made tests that are more comprehensive. We made tests for each phase separately. In each phase, there were 3 regular sizes of the system (128G, 256G, 512G) and 3 modified variants with the same sizes, but with optimizations as stated in 6.4. Each variant has 4 independent test sets inside (male dev, male test, female dev, female test).

Size (G)	EER (%)				**EER performance optimized(%)**			
	Male		Female		Male		Female	
	test	dev	test	dev	test	dev	test	dev
128	18.91	19.65	20.05	18.79	19.97	20.40	21.40	19.34
256	18.12	18.11	19.88	18.60	19.52	19.76	21.95	20.20
512	17.42	17.09	18.85	17.67	18.39	18.57	21.58	19.40

Table 7.2: Results from MOBIO Database from Phase I

Size (G)	EER (%)				**EER performance optimized(%)**			
	Male		Female		Male		Female	
	test	dev	test	dev	test	dev	test	dev
128	15.86	16.27	16.61	14.13	16.86	17.98	17.41	15.13
256	15.10	14.80	17.51	14.88	16.69	15.87	18.78	15.71
512	14.67	14.41	17.12	14.54	15.95	15.79	18.04	15.50

Table 7.3: Results from MOBIO Database from Phase II

7.3 Results Commentary and Comparison with Laboratory System

We can see that the results obtained are far below the performance of BUT best system on NIST data. Depending on the set we get about 15% - 20%. This was nonetheless expected, as length of utterances was too short to successfully score them. In addition, the data sources were problematic - especially type of the microphones (distant microphone). Finally, the whole system was dramatically downscaled to fit in the mobile phone.

In production non-mobile system, which uses 2048G, with iVector normalization, LDA reduction and PLDA as described in [9], we can get EER to the value 2.06% on NIST 2010 dataset. However, this system is not suitable for the mobile device now.

Chapter 8

Conclusion

In this work, we managed to port our „cutting edge“ SID system to the mobile phone. It was included in two demo applications presented at Biometrics 2010 in London. MOBIO project was successfully finished in December 2010.

8.1 Range of Work Conducted in This Project

First, I had to learn how to work with the target platform and mobile phone. This includes also set-up of the development environment for the Nokia N900. Then it was necessary to simplify the baseline SID system to the proper size to fit it into the mobile phone. I also had to design and implement the optimization changes in the system. They incorporated both code and algorithmic optimizations. To get valid results for the precision measuring and fusion of the SID system with other parts of the MOBIO project, it was necessary to run proper tests of the system when the optimizations were done. Finally, I had to prepare and test the system in the mobile phone from the user point of view.

8.2 Future Work Perspectives

At this point, it is possible to reimplement this SID system to almost every mobile device with enough processing power. In my point of view, we should focus on the reimplementation to the Android platform. There will be a few obstacles, primarily because of the Java VM Dalvik, which has different architecture than Maemo OS. The whole framework should be then rewritten to the Java and only crucial parts and BS-CORE library should be kept in the native C code. The other possible direction may be the implementation of the SID to the small SoC (System on the Chip) box with strong DSP optimization and ARM processor utilization.

8.3 Personal Conclusion

This project, MOBIO, provided me with significant new experience resulting from its large scope and its international nature. This project allowed me to improve my skills in Speaker Identification systems, especially of how these systems work inside. It also gives me an opportunity to meet people from the other universities and companies working in the face and speech recognition field. I was able to work with international team of specialists starting from the people with great implementation knowledge to recognition experts and

up to the PR managers. I hope I will be able to use experience from this project in my future work.

Appendix A

SpeakerVerification.h

```
1  #ifndef __MOBIO_SPEAKERVERIFICATIONLIB_H__
2  #define __MOBIO_SPEAKERVERIFICATIONLIB_H__
3
4
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8  #include <math.h>
9  #include "MOBIO_TypeDefs.h"
10 #define BITSPERSAMPLE 16
11 #define CHANNELS 1
12 #define MIN_SV_FRAMES 150
13 #define MAX_SV_FRAMES 500
14
15 class MOBIO_SpeakerVerificationLib {
16 public:
17     /** Create speaker verification object
18     @param aFs  Sampling rate
19     @param aLengthInMs  Buffen length in milliseconds
20      */
21     MOBIO_SpeakerVerificationLib(int aFs, int aLengthInMs);
22
23     /** Destructor relases all the resources allocated during initialization
           of the method.
24      */
25     ~MOBIO_SpeakerVerificationLib();
26
27     /** Apply the speaker verification for current input audio buffer  (
           against the model loaded with the MOBIO_loadSVspeakerID)
28     @param aAdudioBuffer Pointer to input audio containing _iFrameSize of
           samples
29     @param aNumberOfSamples Number of audio samples in current buffer
30     @return 1 if verication was succesfully executed, otherwise 0
31      */
32     int MOBIO_getSVresult(short *aAudioBuffer, int aNumberOfSamples);
33
34     /** Get the score of the verification
35     @return The score of the latest verification
36      */
37     float MOBIO_getSVscore();
38
39     /** Set number of frames used in enrollment
```

```
40      @param aCount value to set
41       */
42      void MOBIO_setSVmaxEnrollFrames(int aCount);
43
44      /** Get the number of enrollment frames
45      @return Number of enrollment frames
46       */
47      int MOBIO_getSVmaxEnrollFrames() {
48          return maxEnrollFrames;
49      }
50
51      /** Create or update the speaker model
52      @param aAdudioBuffer Pointer to input audio containing _iLengthInMs
            milliseconds of samples
53      @param aNumberOfSamples Number of audio samples in current buffer
54      @param aID The ID of the person
55      @param aUpdate 1 if the model is updated, 2 if last addition to model,
            otherwise new model is created
56      @return 1 if verification is reaady, -1 if error occurs and 0 if the
            results are not ready yet (still more frames are needed)
57       */
58      int MOBIO_enrollSV(short *aAudioBuffer, int aNumberOfSamples, char *aID,
            int aUpdate);
59
60      /** Load speaker id data (features etc.) for the specific person ID
61      @param aID The id of the person to be loaded (can be filename)
62      @param 1 if the model was succesfully loaded, otherwise 0
63       */
64      int MOBIO_loadSVspeakerID(char *aID);
65
66
67
68  private:
69      /** The sampling rate */
70      int _iFs;
71      /** The lenght of the frame in samples */
72      int _iFrameSize;
73      /** The length of the audio buffer */
74      int _iLengthInMs;
75      /** The score of speaker verification */
76      float _fScore;
77      /** Flag to determine if the initialization was succesful */
78      bool _bInitialized;
79
80      // Other lib specific member variables
81      // Please notice that you should not allocate memory in the processing
            methods!
82      // Consider to declare class members here and allocate everything during
            initialization!
83
84
85  };
86
87  #endif  /* __MOBIO_SPEAKERVERIFICATIONLIB_H__ */
```

Listing A.1: Example of code implemented in NEON FPU intrinsics

Bibliography

[1] Niko Brummer, Lukáš Burget, Patrick Kenny, Pavel Matějka, Edward Villiers de, Martin Karafiát, Marcel Kockmann, Ondřej Glembek, Oldřich Plchot, Doris Baum, and Mohammed Senoussauoi. Abc system description for nist sre 2010. In *Proc. NIST 2010 Speaker Recognition Evaluation*, pages 1–20. National Institute of Standards and Technology, 2010.

[2] N. Dehak, P. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front–end factor analysis for speaker verification. In *IEEE Trans. on Audio, Speech and Lang. Process.*, 2010.

[3] P. Kenny. Joint factor analysis of speaker and session variability : Theory and algorithms - technical report CRIM-06/08-13. Montreal, CRIM, 2005, 2005.

[4] P. Kenny. Bayesian speaker verification with heavy–tailed priors. keynote presentation, Proc. of Odyssey 2010, June 2010.

[5] P. Kenny, P. Ouellet, N. Dehak, V. Gupta, and P. Dumouchel. A study of inter-speaker variability in speaker verification. *IEEE Trans. Audio, Speech and Language Processing*, 16(5):980–988, July 2008.

[6] Craig S Martin, Alvin F. / Greenberg. The nist 2010 speaker recognition evaluation. In *INTERSPEECH-2010*.

[7] Pavel Matějka Martin Karafiát Patrick Kenny Ondřej Glembek, Lukáš Burget. Simplification and optimization of i-vector extraction. In *ICASSP 2011*.

[8] Mobio A. Hadid (OULU) and C. McCool (IDIAP). Description of mobio database. `https://www.idiap.ch/dataset/mobio/description-1`.

[9] Fabio Castaldo M.J. Alam Oldřich Plchot Patrick Kenny Lukáš Burget Pavel Matějka, Ondřej Glembek and Jan Černocký. Full-covariance ubm and heavy-tailed plda in i-vector speaker verification. In *ICASSP 2011*.

[10] S. J. D. Prince and J. H. Elder. Probabilistic linear discriminant analysis for inferences about identity. In *11th International Conference on Computer Vision*, pages 1–8, 2007.

[11] D. A. Reynolds. Automatic speaker recognition using gaussian mixture speaker models. In *MIT Lincoln Laboratory Journal*, volume 8, pages 173–192, 1995.

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

[13] P. Tresadern (UMAN) S. Marcel (IDIAP), P. Matějka (BUT). Mobio project - deliverable 6.4. http://www.mobioproject.org/project/deliverables.

[14] P. Schwarz, P. Matějka, and J. Černocký. Hierarchical structures of neural networks for phoneme recognition. pages 325–328, Toulouse, France, May 2006.

[15] Giorgio Zoia. Mobio project - deliverable 6.2. http://www.mobioproject.org/project/deliverables.