UNIVERSITY "POLITEHNICA" OF BUCHAREST
FACULTY OF ELECTRONICS, TELECOMMUNICATION AND INFORMATION TECHNOLOGY

Speaker Recognition on Embedded System (on NAO Robot)

Diploma Thesis

submitted in partial fulfillment of the requirements for the **Degree of Engineer** in the domain **Electronics and Telecommunications**, study program **Technology and Telecommunication Systems**

Thesis Advisor
Prof. Ph. D. Corneliu BURILEANU

Student
Rebeca-Graţiela Predescu

Bucharest
2018

University "Politehnica" of Bucharest
Faculty of Electronics, Telecommunications and Information Technology
Department **Tc**

Anexa 1

DIPLOMA THESIS

of student **PREDESCU A. Rebeca-Graţiela , 441G**

1. Thesis title: Speaker Recognition on Embedded System (on Nao Robot)

2. The student's original contribution will consist of (not including the documentation part):
The purpose of this thesis is developing a program that allows the Nao robot to recognize some people based on their voice's characteristics. The first step towards achieving this goal is the voice signal capturing, the next step being the filtering of the noise. After that, all necessary computations are done such that the robot is able to uniquely identify a speaker whose voice characteristics it has previously learned about. The last part is represented by the decision making of the Nao robot and the appropriate response it gives to indicate the speaker.

3. Pre-existent materials and resources used for the project's development:
Nao robot, Matlab

4. The project is based on knowledge mainly from the following 3-4 courses:
Microcontrollers, Microprocessor Architecture, Object-Oriented Programming,Digital Signal Processing

5. The Intellectual Property upon the project belongs to: the student

6. Thesis registration date: 2017-11-29 14:02:33

Thesis advisor(s),
Prof. dr. ing. Corneliu BURILEANU
signature:.....

Student,
signature:.....

Departament director,
Conf. dr. ing. Eduard POPOVICI
signature:.....

Dean,
Prof. dr. ing. Cristian NEGRESCU
signature:.....

Validation code: **83120e8ca8**

STATEMENT OF ACADEMIC HONESTY

I hereby declare that the thesis "*Speaker Recognition on Embedded Systems (on Nao Robot)*", submitted to the Faculty of Electronics, Telecommunications and Information Technology in partial fulfillment of the requirements for the degree of Engineer of Science in the domain *Technology and Telecommunication Systems*, study program *Telecommunication and Information Technology*, is written by myself and was never before submitted to any other faculty or higher learning institution in Romania or any other country.

I declare that all information sources sources I used, including the ones I found on the Internet, are properly cited in the thesis as bibliographical references. Text fragments cited "as is" or translated from other languages are written between quotes and are referenced to the source. Reformulation using different words of a certain text is also properly referenced. I understand plagiarism constitutes an offence punishable by law.

I declare that all the results I present as coming from simulations or measurements I performed, together with the procedures used to obtain them, are real and indeed come from the respective simulations or measurements. I understand that data faking is an offence punishable according to the University regulations.

Bucharest, 2018

Rebeca-Grațiela Predescu

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ABS-PC : Acrylonitrile Butanide Styrene Polycarbonate
ASK : Autism Solutions for Kids
ASR: Automatic Speech Recognition

CPU : Central Processing Unit

DFOV : Dual Field of View
DFT: Discrete Fourier Transform

EM: Expectation Maximization

FSR : Force Sensitive Resistors

GMM: Gaussian Mixture Model

HMM: Hidden Markov Model

LED : Light Emitting Diode

MFCC: Mel Frequency Cepstral Coefficients
ML: Maximum Likelihood

PA-66 : PolyAmide 66
PDF: Probability Density Function

RAM : Random Access Memory
RISC : Reduced Instruction Set Controller

SDHC : Secure Digital High Capacity

UBM: Universal Background Model
USB : Universal Serial Bus

WSS: Wide-Sense Stationary

CHAPTER 1
INTRODUCTION

1.1 THESIS MOTIVATION

In the last few decades, more and more people suffer from different conditions that decrease drastically their life quality. In a desire to help them integrate in the society, several measures should be taken to ease their life, among which are also some techniques developed with the purpose of correcting certain unwanted behavior traits.

The core of this project is represented by the NAO robot, created by Aldebaran Robotics especially to be used in therapy. For a child that needs to be attracted by the whole activity performed during therapy to learn basic human behavior traits, an appealing method should be applied. In this context, NAO is the ideal candidate, as it can be used to help patients learn different words, recognize patterns, make certain movements. Human intervention is essential at present, but more and more autonomy for the robot is desired.

1.2 MAIN OBJECTIVE

In this context, the thesis aims to create an autonomous system that is able to interact with people using their voice. The system works in Romanian language and the voice recognition software that was developed is essential to increase the autonomy of the robot. In addition to similar projects developed already by other students, my thesis comes with the advantage of having all functions directly implemented on the robot, thus eliminating the need for an internet connection that would have been necessary to send files to and from the server.

Embedded programming presents a series of disadvantages, the most important being the limited amount of resources, that, in this case, are only the ones that are available on the robot itself. Another constraint is represented by the number of people it can recognize, because it is desired to have real time processing, and the response time to be as short as possible. Despite these drawbacks, embedded programming is the optimal solution in this case, as the time

required to obtain a result locally on the machine is much smaller than the one needed to send information to the server and receive back the processed data.

The steps required to reach the proposed objective are represented below:

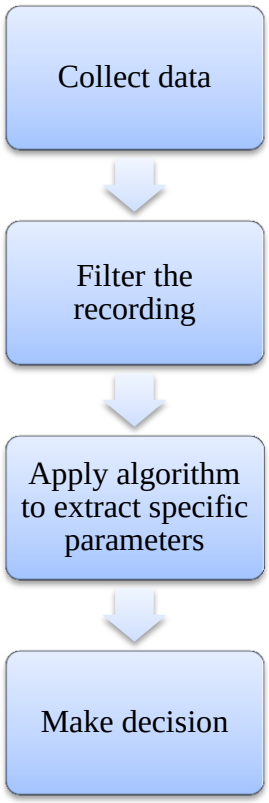

Figure 1.1 Implementation Steps

1.3 SPECIFIC OBJECTIVES

The objectives this thesis proposes are, as follows:

- Collecting data corresponding to several users that will be identified by the robot. Data is represented by several recordings of the voice of the people that are to be recognized and it is gathered by the robot itself.
- Extracting parameters that define the voice of a single person and map them in a database, thus being able to uniquely identify people from the restricted set that was imposed.
- Update the database such that the model corresponding to each user gives minimum errors.
- Giving a message to check if the robot recognized the person or not.
- Study the effects of noise on the overall results.

- Study the variation of the accuracy with the various adjustable parameters in order to make the best complexity – results compromise.

This thesis is organized in six chapters. *Chapter 1* presents the motivation along with the objectives of the thesis. In *Chapter 2* a detailed presentation of the hardware and software technologies used in the implementation of the project is being made. *Chapter 3* describes the filtering method I applied to the information collected by NAO. In *Chapter 4*, the voice recognition algorithm is presented, details regarding each step being given. *Chapter 5* presents the tests made, along with the results obtained for each of the studied cases. In *Chapter 6*, the main conclusions of the thesis are drawn and the contributions the author brought to the project are emphasized, along with some further possible directions.

CHAPTER 2
HUMANOID ROBOTS

2.1 GENERAL ASPECTS

A humanoid robot is a robot whose appearance is based on that of the human body. The most important physical characteristics such a robot has are the head, a torso, two hands and two legs, with some exceptions regarding earlier versions, when only the superior part of the body was built. Usually, the head resembles that of a person, having two eyes, a mouth and ears that map some of its functionalities. [1]

During the last few years, the existence of such robots became more and more a necessity, since many tasks could be easily carried on by them. The goal is to make the robot autonomous, so that it can work without any human help. This way, it can complete complex tasks; it can communicate, learn from people and interact with them.

2.2 APPLICATIONS OF HUMANOID ROBOTS

Since their apparition, many applications have been found for humanoid robots. They could be successfully used for spatial applications, therapy, quenching flames and other rescue missions, help with some chores around the house, and so on.

Even though many possible functions are still in research stage, promising results are obtained in the laboratory. Challenges may occur due to the fact that every ability the robot is expected to have needs to be carefully programmed and tested. Also, the desire to have an autonomous system imposes some tougher requirements on the software characteristics.

The term “autonomous” refers to the ability of the robot to perform tasks without being controlled by humans. The degree of autonomy is increased by self-learning and safe-developing. [2]

2.3 NAO

Developed in 2006, NAO is the first humanoid robot from Aldebaran and it reached the fifth generation, each adding more functionalities to the existing ones. It was designed as a studying tool, to help young students learn to count, tell a story, create a choreography or even learn how to program the robot. Regarding specialized education, the ASK program comes with a solution that includes the robot NAO in therapy, having some applications especially written to meet the needs of autistic children. [3]

Figure 2.3.1 NAO Robot [4]

2.3.1 *General Features*

The 574 mm height, along with its friendly complexion, make NAO an agreeable presence around people. It weighs only 5.4 kilograms, which makes it easy to transport and manipulate. The ratio height/weight was chosen such that to ensure motion stability. The material is a combination of ABS-PC and PA-66 that offers flexibility without losing strength and also good thermic protection. [5]

2.3.2 *Resources*

NAO is equipped with a Lithium-Ion battery, having the nominal capacity of 2.25Ah. The charging duration is less than three hours and the autonomy is of about 60 minutes. The robot can be used while it is plugged in. [5]

NAO has a single nucleus processor, ATOM Z530 that is usually utilized for mobile devices due to its low energy consumption. The CPU has the clock of 1.6 GHz. The 1GB RAM memory is one of the limitations presented by the robot for real-time applications. The Flash memory is of 2GB. An 8GB Micro SDHC can be used at maximum. At the torso level, another processor is used, with the purpose of controlling the actuators. ARM7TDMI is a 32 bit, RISC microcontroller. [5]

Regarding the connectivity, it can be done via Ethernet or Wi-Fi. The Ethernet port can be accessed on the back of the head with a RJ45 jack. The speeds supported are 100Mbps, 1000Mbps or 1Gbps. To update the system of the robot, an Arduino device, Kinect or Asus 3D sensor can be connected through the USB port placed at the back of the head. [5]

To be able to provide audio interaction, NAO is endowed with two loudspeakers, placed in its ears. Four microphones allow the stereo recording of sounds, with a maximum 44.1 kHz sampling frequency. The frequency range for the microphones is between 300Hz and 8 kHz and the sensitivity is 20mV/Pa +/- 3dB at 1 kHz.

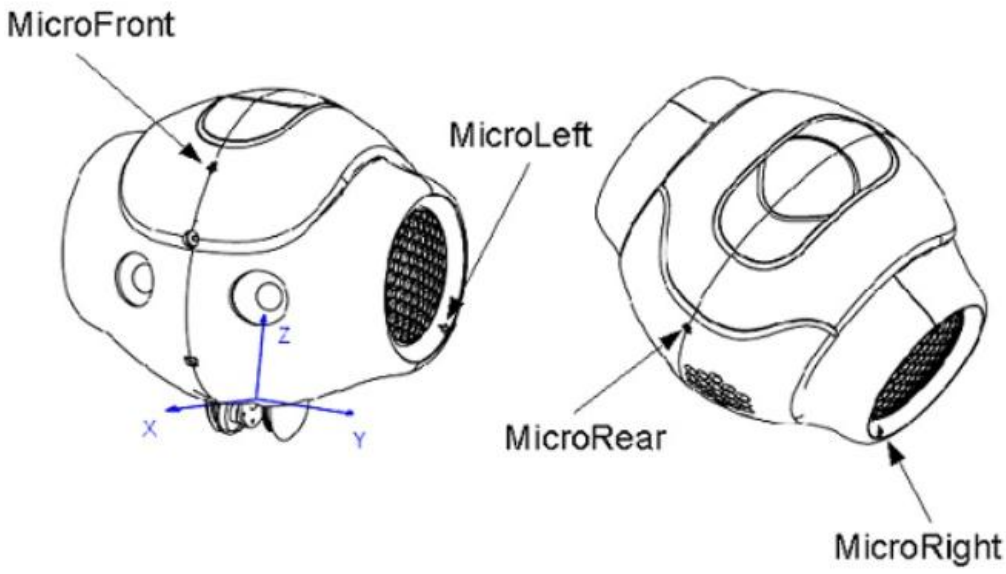

Figure 2.3.2.1 Microphones' Location [5]

To process images coming from the medium, NAO has two cameras on its forehead. They are identical and provide a resolution up to 1280x960 at 30 frames per second. The cameras are used both to identify objects in the robot's vision field and to help NAO avoid obstacles.

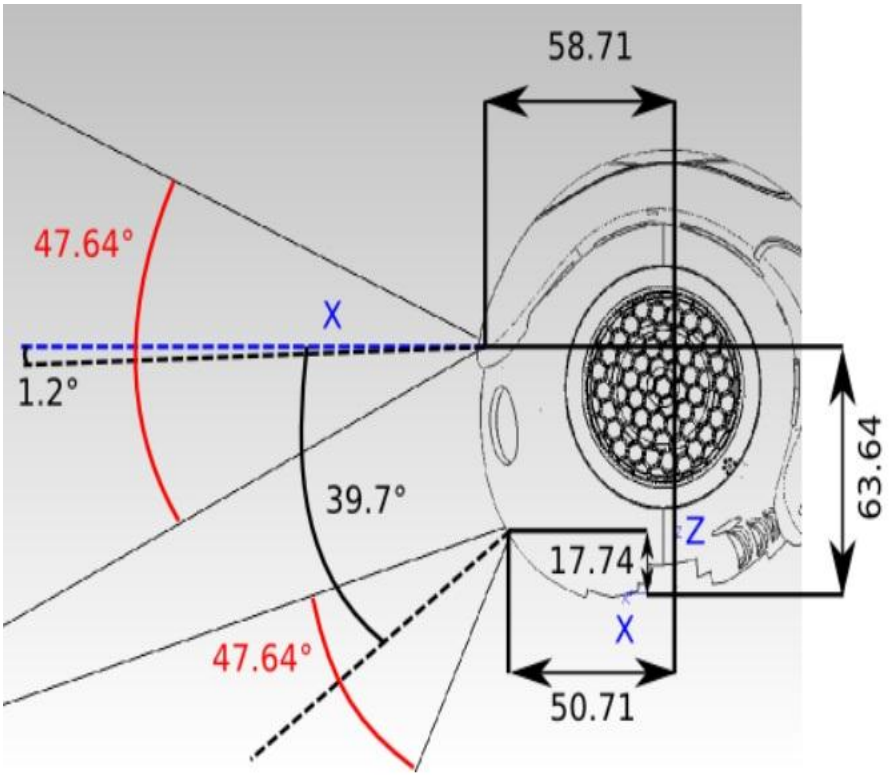

Figure 2.3.2.2 Cameras Used for Object Identification [5]

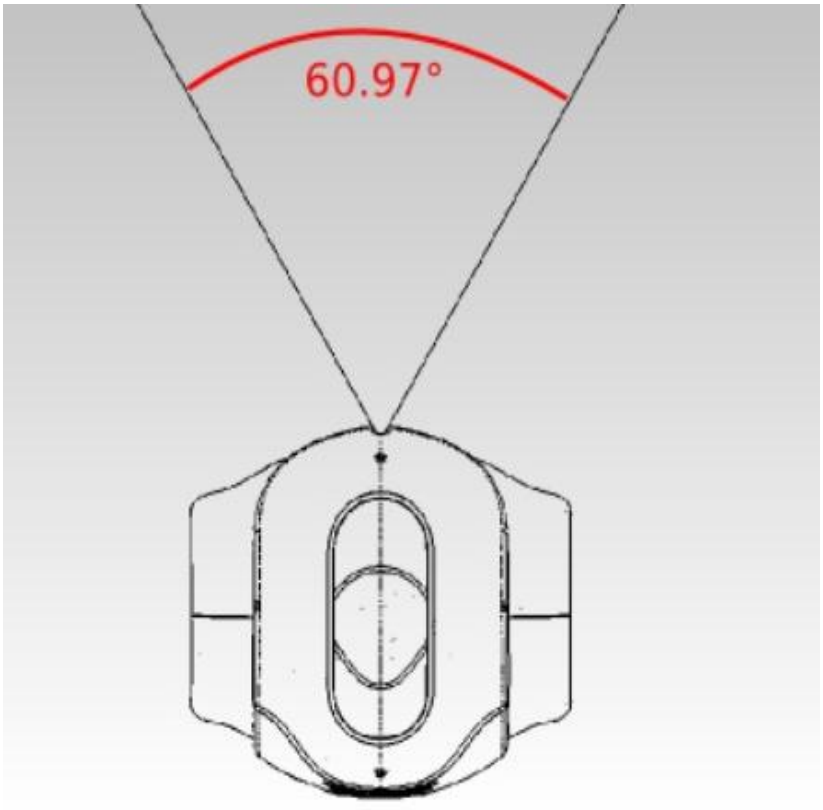

Figure 2.3.2.3 Camera Used to Detect Obstacles [5]

The cameras are of type SOC Image Sensor, model MT9M114. The image array is defined through the resolution of 1.22 Mp, optical format 1/6 inch and active pixels of 1288x968. Regarding the sensitivity, the pixel size is 1.9μm*1.9μm. The dynamic range is of 70dB, while the signal-to-noise ratio is, at maximum, 37dB. The field of view is 70.6˚DFOV, having 60.9˚the horizontal field of view and 47.6˚the vertical one. The focus is of fixed type and its range starts at 30 cm. The cameras output 1280x960 at 30 frames per second. The shutter is of type Electronic Rolling Shutter. [5]

NAO has many LEDs that make it pleasant and also can be used to mark some aspects regarding the functionality. For example, when the eyes turn red, it means that the battery is low. The LEDs are placed according to the figure:

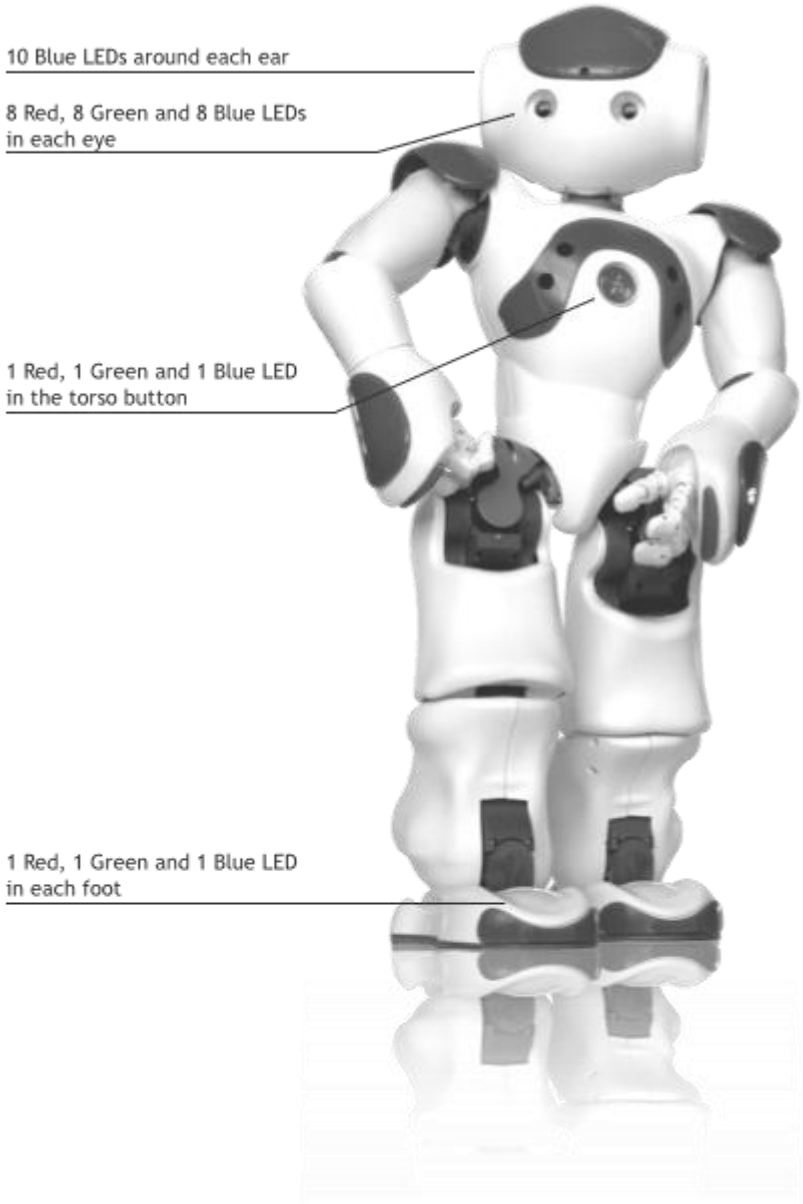

Figure 2.3.2.4 LED Positions [6]

The LEDs in the ears are all blue. The eyes, chest and feet LEDs are red, green and blue, which properly combined give the whole color spectrum. Also, the light intensity can be varied between 0 and 100%.

Force sensitive resistances are the sensors that measure the resistance change according to the pressure applied. They are placed on the robot's feet. [5]

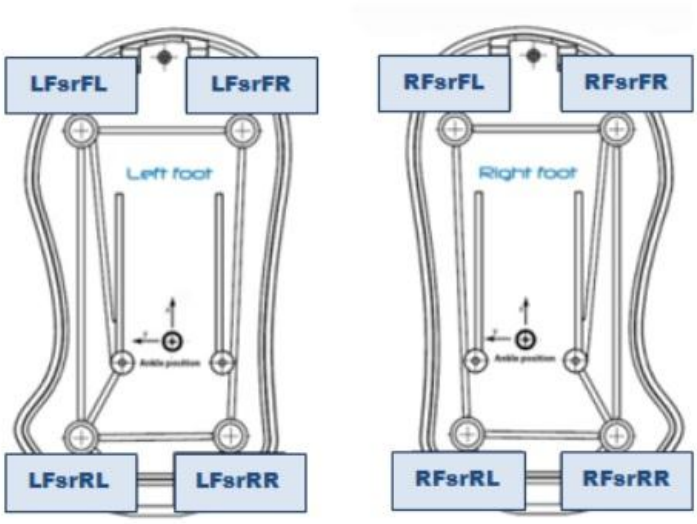

Figure 2.3.2.5 FSR Sensors [5]

The inertial unit is placed in the robot's torso and has its own processor. It consists of 2 axis gyrometers and one three-axis accelerometer. The precision of the gyrometers is 5% and their angular speed of 500˚/s. For the accelerometer, the precision is 1%. The accelerometer gives the angle of the torso in static mode and is considered the reference. When motion is detected, the output angle is computed using the gyrometers. [5]

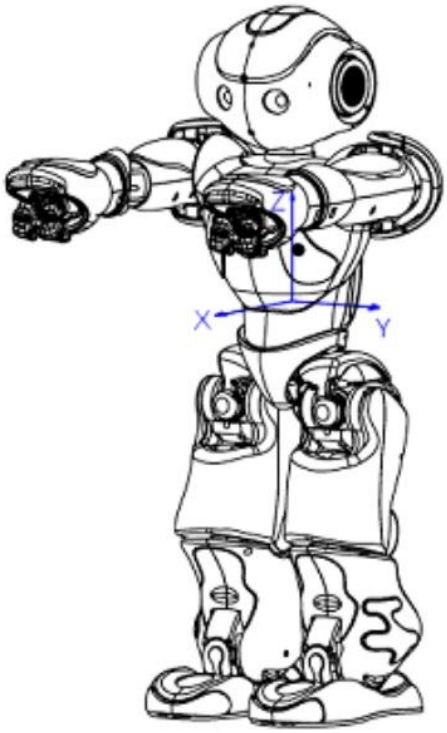

Figure 2.3.2.6 Inertial Unit [5]

To facilitate the motion, NAO is equipped with 2 ultrasonic sensors. They allow the estimation of distances to reach an obstacle. For distances smaller than 25 cm, the robot only senses an object, but cannot give supplementary information about its exact position. [5]

The characteristics of the sonars are given in the following table:

Frequency	40 kHz
Sensitivity	-86 dB
Resolution	1 cm
Detection Range	0.25 m – 2.55 m
Effective Cone	60˚

Table 2.3.2.1 Sonars' Characteristics [5]

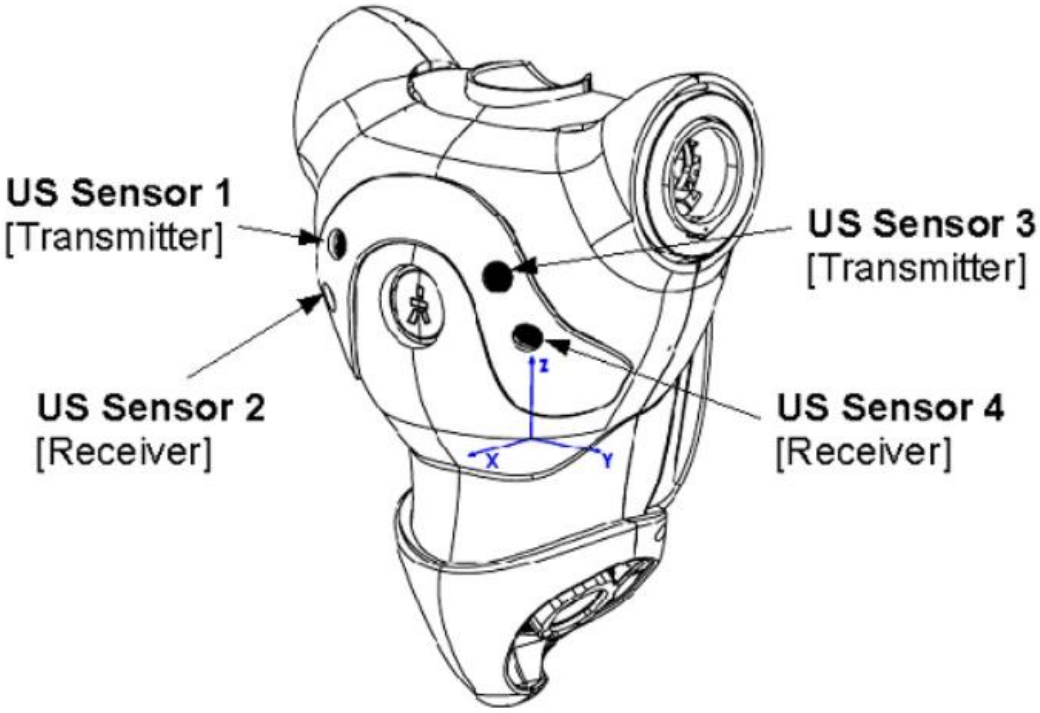

Figure 2.3.2.7 Sonars' Position [5]

The joint position sensors have 12 bit precision, that give a precision of about 0.1˚.

The robot also has 3 capacitive sensors on top of the head that can be programmed to trigger different actions. A chest button is used to turn on and off NAO. The 3 sensors on each hand and the 2 bumpers on the tip of each foot have protection purposes: they prevent the robot from hitting its hands and feet on foreign objects.

NAO has 25 motors, one for each joint. They give the robot all liberty needed to make its movements as natural as possible. The 25 degrees of freedom are divided into 11 for the inferior part and 14 for the superior one, including the head. [5]

Part of robot's body	Degrees of freedom
Head	2
Arm	5 for each
Torso	1
Leg	5 for each
Hand	1 for each

Table 2.3.2.2 Degrees of freedom [5]

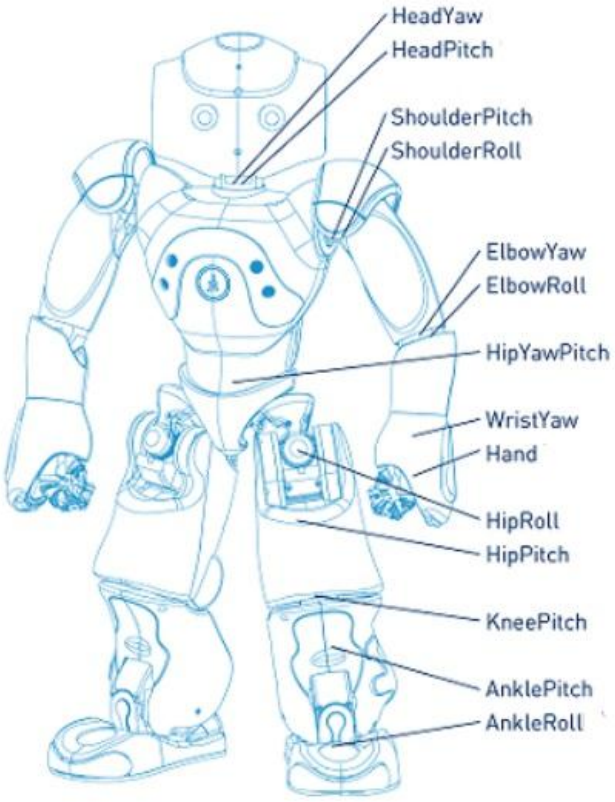

Figure 2.3.2.8 Motors' Position [5]

The robot has three types of motors, each having different characteristics and advantages. The carbon brush actuators have a reduced cost and the speed can be configured by the user, which is a major advantage when programming movements. [5]

	Motor Type 1	Motor Type 2	Motor Type 3
Model	22NT82213P	17N88208E	16GT83210E
No load speed	8 300 rpm ±10%	8 400 rpm ±12%	10 700 rpm ±10%
Stall torque	68 mNm ±8%	9,4 mNm ±8%	14,3 mNm ±8%
Nominal torque	16.1 mNm	4.9 mNm	6.2 mNm

Table 2.3.2.3 Motor Types [5]

For the legs, motors of type 1 are used, as they are the most powerful from the three. Type 2 are used for the hand joints and type 3 motors for the arms and head.

CHAPTER 3
WIENER FILTERING

3.1 GENERAL ASPECTS

The useful speech signal is usually affected by noise, which compromises the results obtained after processing. In order to minimize the effect it has on the useful signal, some filtering methods were developed, that help enhance the speech signal.

The Wiener filtering method is one of the most used techniques for signal enhancement. It is used to produce an estimate of the desired signal, by having as inputs the noisy signal and assuming that the noise is additive. It minimizes the mean square error between the desired result and the estimated one. To determine the filter coefficients, the spectral properties of the original, compromised signal and of the noise should be known. Also, the original signal and the noise are considered stationary, linear stochastic processes. [7]

3.2 RANDOM SIGNALS AND SPECTRAL DENSITIES [8]

Let *x* be a discrete time signal, defined as:

$$x = (\ldots\ldots, x_{-1}, x_0, x_1, \ldots\ldots);$$

Its Fourier and Z transforms are

$$X(e^{j\omega}) = \sum_{k=-\infty}^{\infty} x_k e^{-j\omega k} \qquad , \omega \in [-\pi;\ \pi]$$

$$X(z) = \sum_{k=-\infty}^{\infty} x_k z^{-k} \qquad , z \in D_z$$

The correlation function is defined as follows:

$$R_x(k,m) = E(x_k x_m^T)$$

A signal is wide-sense stationary (WSS) if the following conditions are met:

- the mean of the signal is constant, time invariant

$$E[x_k] = E[x_0]$$

- the autocorrelation function does not depend on the absolute time, but only on the time difference between the two moments when it is calculated

$$R_x(m,k) = R_x(m-k)$$

It can be proven that, for WSS signals, the autocorrelation function is even, that is:

$$R_x(-k) = R_x(k)^T$$

The power spectral density of a WSS signal is:

$$S_x(e^{j\omega}) = \sum_{k=-\infty}^{\infty} R_x(k) e^{-j\omega k} \qquad , \omega \in [-\pi;\ \pi]$$

Two WSS stationary processes are joint-WSS if:

$$R_{xy}(k,m) = E(x_k y_m^T) = R_{xy}(m-k)$$

Their power spectral density:

$$S_{xy}(e^{j\omega}) = S_{yx}(e^{-j\omega})^T$$

3.3 LINEAR TIME-INVARIANT SYSTEMS

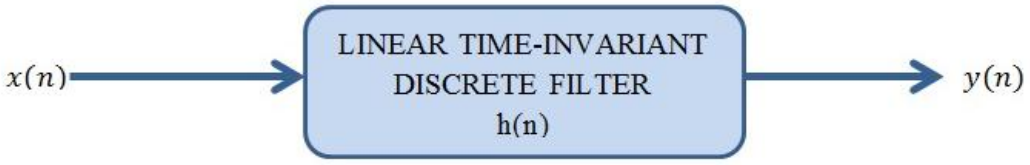

Figure 3.3.1 Linear Time-Invariant Discrete Filter

Being given the system from Figure 3.3.1, its output is defined as the impulse response of the filter:

$$y(n) = \sum_{k=-\infty}^{\infty} h(n-k)x(k)$$

A linear time-invariant discrete system is stable if its impulse response function is absolutely summable, that is:

$$\sum_{n=-\infty}^{\infty} |h(n)| < \infty$$

For stable systems, the output in frequency domain is:

$$Y(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega})$$

And, in the z domain,

$$Y(z) = H(z)X(z)$$

The system is stable if all poles of H(z) are inside the unity circle.

3.4 WIENER FILTER'S COEFFICIENTS [9]

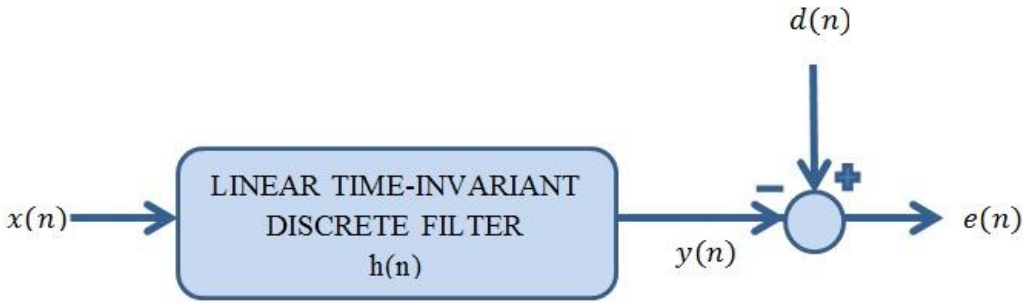

Figure 3.4.1 Wiener Filter

The coefficients of the Wiener filter will be determined with the help of the above diagram, where *x(n)* denotes the input samples of the signal, *d(n)* is the signal desired to be obtained, *y(n)* is the output of the filter, *h(n)* the impulse response function and *e(n)* represents the error signal. The Wiener filtering works with stationary random processes with zero mean.

Considering real, causal signals at the input, the impulse response function can be written as:

$$h(n) = \sum_{k=0}^{M-1} w_k \delta(n-k)$$

where w_k denotes the filter's coefficients, M is the length of the filter and δ(n) is the Dirac distribution, defined as follows:

$$\delta(k) = \begin{cases} 1, & k = 0 \\ 0, & k \neq 0 \end{cases}$$

The output of the filter, y(n), will be:

$$y(n) = \sum_{k=0}^{M-1} w_k x(n-k)$$

The Wiener filters realize the optimization in the mean-square sense, that is by minimizing the mean-square error that appears between the output of the filter and the desired signal. In consequence, the cost function is defined:

$$J_{MS} = E\{|e(n)|^2\} = ||e(n)||^2 = \langle e, e \rangle \, \epsilon \, R$$

So, the goal is to find the filter's coefficients for which the cost function is minimal.
In order to do so, the expression of the error function is written:

$$e(n) = d(n) - y(n) = d(n) - \sum_{k=0}^{M-1} w_k x(n-k)$$

The cost function thus becomes:

$$\begin{aligned} J_{MS} = E\{e(n)^2\} &= E\{[d(n) - \sum_{k=0}^{M-1} w_k x(n-k)]^2\} \\ &= E\{d(n)^2\} - E\{\, d(n) \sum_{k=0}^{M-1} w_k x(n-k) \,\} - E\{ \sum_{k=0}^{M-1} w_k x(n-k) \; d(n) \,\} \\ &+ E\{ \sum_{k=0}^{M-1} w_k x(n-k) \sum_{i=0}^{M-1} w_i x(n-i) \,\} \end{aligned}$$

The correlation function is:

$$r_{xy}(k) = E\{x(n-k)y(n)\} = E\{x(n)y(n-k)\}$$

The coefficients of the filter are constants and real, so the mean operator does not affect them. Taking these into consideration, the cost function becomes:

$$\begin{aligned} J_{MS} = E\{d(n)^2\} & \\ &- \sum_{k=0}^{M-1} w_k E\{d(n)x(n-k)\} \\ &- \sum_{k=0}^{M-1} w_k E\{x(n-k)d(n)\} + \sum_{k=0}^{M-1} \sum_{i=0}^{M-1} w_k w_i \, E\{x(n-k)x(n-i)\} \\ &= E\{d(n)^2\} - \sum_{k=0}^{M-1} w_k r_{dx}(k) - \sum_{k=0}^{M-1} w_k r_{xd}(-k) + \sum_{k=0}^{M-1} \sum_{i=0}^{M-1} w_k w_i \, r_{xx}(i-k) \end{aligned}$$

With $r_{dx}(k)$ was denoted the correlation function between the input signal and the desired one and with $r_{xx}(k)$ the autocorrelation function of the input signal.

For real valued processes,

$$r_{xy}(k) = r_{yx}(-k)$$

$$J_{MS} = E\{d(n)^2\} - 2\sum_{k=0}^{M-1} w_k r_{xd}(-k) + \sum_{k=0}^{M-1}\sum_{i=0}^{M-1} w_k w_i\, r_{xx}(i-k)$$

To write in a matrix form, the following vectors are defined:

$$\boldsymbol{w} = [w_o\, w_1\, \ldots . w_{M-1}]^T$$
$$\boldsymbol{x}(n) = [x(n)\;\; x(n-1)\;\; \ldots .\;\; x(n-M+1)]^T$$
$$\boldsymbol{p} = [r_{xd}(0)\;\; r_{xd}(-1) \ldots .\;\; r_{xd}(-M+1)]^T$$

$$\mathbf{R} = \mathrm{E}\{\boldsymbol{x}(n)\boldsymbol{x}(n)^T\} = \begin{bmatrix} r_{xx}(0) & \cdots & r_{xx}(-M+1) \\ \vdots & \ddots & \vdots \\ r_{xx}(M-1) & \cdots & r_{xx}(0) \end{bmatrix}$$

The cost function in matrix form is:

$$J_{MS} = E\{d(n)^2\} - 2\boldsymbol{w}^T\boldsymbol{p} + \boldsymbol{w}^T\boldsymbol{R}\boldsymbol{w}$$

The minimum is reached when the gradient is zero. The gradient of a scalar function $f(x_1, x_2, \ldots x_N)$ is defined as:

$$\nabla_{\boldsymbol{x}}(f) = \left[\frac{\partial f}{\partial x_1}\;\frac{\partial f}{\partial x_2}\;\ldots .\frac{\partial f}{\partial x_N}\right]^T$$

By applying this expression to the cost function, and observing that *d(n)* does not depend on any *w* coefficient, it results that:

$$\nabla_{\boldsymbol{w}}\{J_{MS}\} = \nabla_{\boldsymbol{w}}\{E\{d(n)^2\} - 2\boldsymbol{w}^T\boldsymbol{p} + \boldsymbol{w}^T\boldsymbol{R}\boldsymbol{w}\} = -\nabla_{\boldsymbol{w}}\{2\boldsymbol{w}^T\boldsymbol{p}\} + \nabla_{\boldsymbol{w}}\{\boldsymbol{w}^T\boldsymbol{R}\boldsymbol{w}\}$$

$$\boldsymbol{w}^T\boldsymbol{p} = [w_o\, w_1\, \ldots . w_{M-1}] \begin{bmatrix} r_{xd}(0) \\ \vdots \\ r_{xd}(-M+1) \end{bmatrix}$$
$$= w_o\, r_{xd}(0) + w_1 r_{xd}(-1) + \cdots + w_{M-1}\, r_{xd}(-M+1)$$

$$\frac{\boldsymbol{\partial}\boldsymbol{w}^T\boldsymbol{p}}{\boldsymbol{\partial} w_k} = r_{xd}(-k)$$

$$\boldsymbol{w}^T\boldsymbol{R}\boldsymbol{w} = [w_o\, w_1\, \ldots . w_{M-1}] \begin{bmatrix} r_{xx}(0) & \cdots & r_{xx}(-M+1) \\ \vdots & \ddots & \vdots \\ r_{xx}(M-1) & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} w_o \\ \vdots \\ w_{M-1} \end{bmatrix}$$
$$= w_o\,[w_o\, r_{xx}(0) + w_1\, r_{xx}(1) + \cdots + w_{M-1}\, r_{xx}(M-1)] + \cdots$$
$$+ w_{M-1}\,[w_o\, r_{xx}(-M+1) + w_1\, r_{xx}(-M+2) + \cdots + w_{M-1}\, r_{xx}(0)]$$

So,

$$\nabla_{\boldsymbol{w}}\{J_{MS}\} = -2\boldsymbol{p} + 2\boldsymbol{R}\boldsymbol{w}$$

To minimize the function,

$$\nabla_{\boldsymbol{w}}\{J_{MS}\} = 0$$

It implies that

$$\boldsymbol{R}\boldsymbol{w} = \boldsymbol{p}$$

The optimal coefficients for the Wiener filter are denoted $\boldsymbol{w}^0$obtained as:

$$\boldsymbol{w}^0 = \boldsymbol{R}^{-1}\boldsymbol{p}$$

For optimal coefficients of the filter, an observation regarding the orthogonality of the input signal with the error can be made.

$$E\{x(n)e(n)\} = E\{x(n)[d(n) - y(n)]\} = E\{x(n)[d(n) - x(n)^T\boldsymbol{w}^0]\}$$
$$= E\{x(n)d(n)\} - E\{x(n)x(n)^T\boldsymbol{w}^0\} = \boldsymbol{p} - \boldsymbol{R}\boldsymbol{w}^0 = 0$$

This infers that the input signal and the error signal are uncorrelated.

3.5 WIENER FILTER USED IN NOISE REDUCTION

The noise reduction is a particular case of Wiener filtering, where the input is the signal affected by noise and the desired signal is the noise. The schematics below describes the principle:

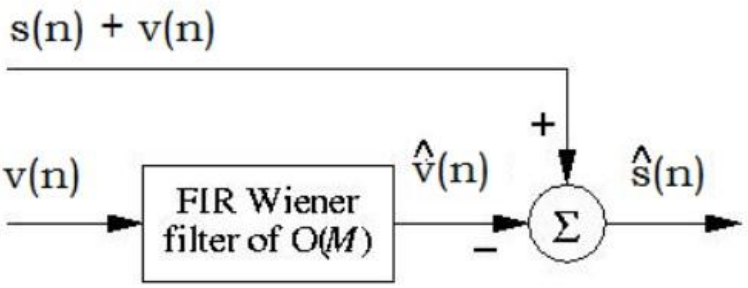

Figure 3.5.1 Noise Reduction Using Wiener Filter [10]

With *s(n)* was denoted the useful signal and with *v(n)* the additive noise; $\hat{v}(n)$ is the noise at the output of the filter, that is subtracted from the noisy signal. The Wiener filter tries to make its output as close as possible to the sum of the desired signal and noise. The correlation between the noise and the useful signal will be zero, while the noise that is added over the signal will be strongly correlated with the noise at the output of the filter. So, when subtracting the two signals, the error signal will be, in fact, the useful signal, without noise.

CHAPTER 4
GMM-UBM

4.1 MARKOV CHAINS

The most important machine learning model in speech processing is the hidden Markov model. In order to be defined, a proper definition for the Markov chain should be given. Markov chains, as well as the hidden Markov models, are extensions of the finite automata, which are defined by states and transitions between them. For a weighted finite-state automaton, each arc that represents a transition between two states is associated with the probability for that transition to occur. So, the probability of a Markov chain to be in a particular state at a given time depends only on the state in which the Markov chain previously was. It can be observed that the probabilities on all arcs leaving a node must be 1. [11]

A Markov chain is a weighted automaton in which the input sequence uniquely determines which states the automaton will go through. It models a class of random processes that incorporate a minimum amount of memory.

Considering $\boldsymbol{X} = X_1, X_2, \dots, X_n$ a sequence of random variables from a discrete alphabet, $O = \{o_1, o_2, \dots, o_M\}$, by applying the Bayes rule,

$$P(X_1, X_2, \dots, X_n) = P(X_1) \prod_{i=2}^{n} P(X_i | X_1^{i-1})$$

where $X_1^{i-1} = X_1, X_2, \dots, X_{i-1}$.

The random variables X form a first-order Markov chain if $P(X_i | X_1^{i-1}) = P(X_i | X_{i-1})$, which means that, for a first-order Markov chain,

$$P(X_1, X_2, \dots, X_n) = P(X_1) \prod_{i=2}^{n} P(X_i | X_{i-1}).$$

This equation is known as the Markov assumption and it uses very little memory to model dynamic data sequences, as the probability of a random variable at a given time depends only on the value at the preceding time, regardless of all other previous values. So, the Markov chain can be used to model stationary signals. [12]

For a Markov chain with *N* distinct states, having the state at time *t* in the Markov chain denoted as s_t, the parameters of the Markov chain are:

$$a_{ij} = P(s_t = j \mid s_{t-1} = i) \quad , 1 \leq i, j \leq N$$

$$\pi_i = P(s_1 = i) \quad , 1 \leq i \leq N$$

With a_{ij} was denoted the transition probability from state *i* to state *j*, while π_i represents the initial probability for the Markov chain to start from state *i*. As it is well known, both transitions' probabilities and initial probabilities are bounded by constraints, that is:

$$\sum_{j=1}^{N} a_{ij} = 1 \quad , 1 \leq i \leq N$$

$$\sum_{i=1}^{N} \pi_i = 1$$

As an example, consider the three-state Markov chain for the Dow Jones Industrial average. At the end of each day, the Dow Jones Industrial average may correspond to one of the states: [12]

- state 1 – up
- state 2 – down
- state 3 – unchanged

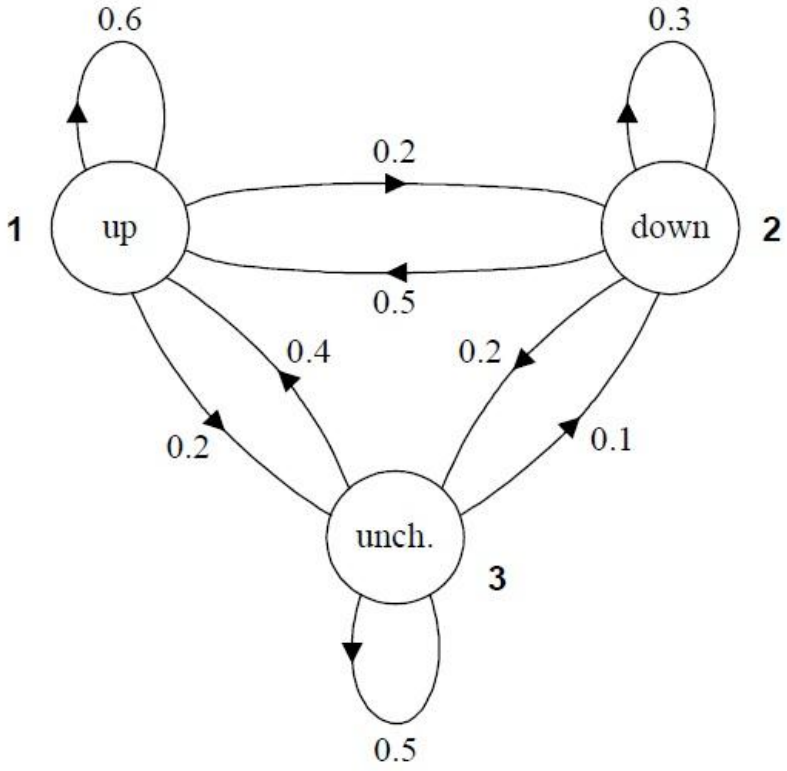

Figure 4.1.1 Markov Chain for Dow Jones Industrial Average [12]

The state-transition probability matrix is:

$$A = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$$

The initial state probability matrix is:

$$\pi = \begin{bmatrix} 0.5 \\ 0.2 \\ 0.3 \end{bmatrix}$$

As it can be seen, both probabilities matrices are defined in accordance with the imposed conditions.

4.2 THE HIDDEN MARKOV MODEL (HMM)

The Markov chain can be used to compute a probability for a sequence of observable events. Sometimes, the events may not be observable. For example, in speech recognition, acoustic events are observable and the presence of hidden words that are the underlying causal source of the acoustics is inferred. A hidden Markov model allows the study of both observed and hidden events that are thought of as the causal factors in the probabilistic model. [11]

A HMM is specified by the following components:

Component	Description
$Q = q_1 q_2 \ldots q_N$	a set of N states
$A = a_{11}a_{12} \ldots a_{1n} \ldots a_{nn}$	a transition probability matrix A, each a_{ij} representing the probability for the transition from state *i* to state *j* to occur
$O = o_1 o_2 \ldots o_T$	a sequence of *T* observations, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of observation likelihoods, each expressing the probability of an observation o_t being generated by the state *i*
q_0, q_F	a special start state and final state that are not associated with observations

Table 4.2.1 HMM Components [11]

As an example, the task proposed by Jason Eisner will be further treated. A climatologist from the year 2799 has to study global warming. He cannot find any records concerning the weather in Baltimore, so he uses the diary of Jason Eisner, where the number of ice creams Jason ate each day is written down. Using these observations, the temperature for each day needs to be estimated. As a simplification, only two kinds of days are considered: cold and hot. So, being given a sequence of observations *O* (the number of ice creams eaten on a given day), the hidden sequence *Q* of weather states should be found. [11]

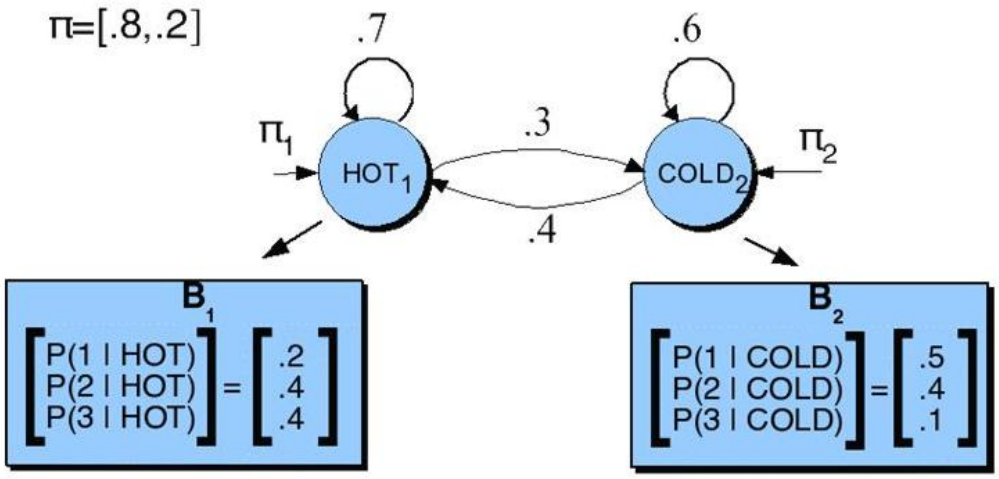

Figure 4.2.1 HMM for the Ice Cream Task [13]

The two hidden states, "hot" and "cold" correspond to hot and cold weather and the observations correspond to the number of ice creams eaten on a given day.

A HMM having non-zero probability of transitioning between any two states is an ergodic HMM (fully connected). [11]

4.3 HMM TRAINING: BAUM-WELCH ALGORITHM

Being given a set of observations *O* and the set of possible states in the HMM, the task is to determine the HMM parameters, that is the *A* (transition probability matrix) and *B* (observation likelihood) matrices.

The algorithm most commonly used to train the HMM is the Baum-Welch algorithm, which is a special case of Expectation Maximization. The advantage of this algorithm is that it allows the training of both the transition probabilities and the emission probabilities of the HMM.

If the simpler case of training a Markov chain is considered, since the states are observed, the model can be run on the observation sequence. As a result, the path taken through the model can be directly observed and so is the state which generated each observation symbol. A Markov chain can be in fact considered a hidden Markov model having all the *b* probabilities equal to 1 for the observed symbol and 0 for all other symbols. So, in this case, only the transmission probabilities *A* should be trained.

To obtain the maximum likelihood estimate of the probability a_{ij} of a transition from state *i* to state *j*, the number of times this transition was taken is counted, denoted by $C(i \rightarrow j)$. Then, the normalization to the number of all transitions from state *i* is performed, and thus a_{ij} is obtained. This can be done only because the states are already known.

$$a_{ij} = \frac{C(i \rightarrow j)}{\sum_{q \epsilon Q} C(i \rightarrow q)}$$

For a HMM, the counts cannot be computed directly from the observation sequence, since the path taken through the machine for a specified input is not known. The Baum-Welch algorithm solves this problem. First of all, the counts are iteratively estimated, that is, starting from an estimate of the transition and observation probabilities, better and better probabilities are obtained. Secondly, the estimated probabilities are obtained by computing the forward probability for an observation and then dividing the probability mass among all the different paths that contributed to this forward probability. [11]

The backward probability β is the probability of seeing the observations from time *t*+1 to the end, given that the current state is *i* at time *t* and the given automation is λ, so:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T \,|q_t = i\,, \lambda\,)$$

It can be computed inductively, as it follows:

- initialization:

$$\beta_t(i) = \; a_{i,F} \quad , 1 \leq i \leq N$$

- recursion:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \quad , 1 \le i \le N \quad , \quad 1 \le t \le T$$

- finish:

$$P(O|\lambda) = \alpha_T(q_F) = \beta_1(0) = \sum_{j=1}^{N} a_{0j} b_j(o_1) \beta_1(j)$$

Now, the estimate transition probability is defined as follows:

$$\hat{a}_{ij} = \frac{expected\ number\ of\ transitions\ from\ state\ i\ to\ state\ j}{expected\ number\ of\ transitions\ from\ state\ i}$$

Being given the observation sequence and the model, the probability of being in state *i* at time *t* and at state *j* at time *t*+1 is defined:

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j \mid O, \lambda)$$

To be able to compute ξ_t, another probability should be first computed, similar to this one but with a different conditioning for *O*:

$$not - quite - \xi_t(i,j) = P(q_t = i, q_{t+1} = j\ , O|\ \lambda) = \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \quad [11]$$

Knowing that:

$$P(X|Y,Z) = \frac{P(X,Y|Z)}{P(Y|Z)}$$

$$P(O|\lambda) = \alpha_T(N) = \beta_T(1) = \sum_{j=1}^{N} \alpha_t(j)\, \beta_t(j)$$

And by introducing these in ξ_t:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(N)}$$

The expected number of transitions from state *i* to state *j* is the sum of ξ over the whole *t* domain. The total expected number of transitions from state *i* is the sum of all transitions coming out of state *i*. Thus, the final expression is:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \sum_{j=1}^{N} \xi_t(i,j)}$$

The probability of a given symbol v_k from the vocabulary *V*, being given a state *j* can be computed using:

$$\hat{b}_j(v_k) = \frac{expected\ number\ of\ times\ in\ state\ j\ and\ observing\ symbol\ v_k}{expected\ number\ of\ times\ in\ state\ j}$$

The probability of being in state *j* at time *t*, $\gamma_t(j)$, is:

$$\gamma_t(j) = P(q_t = j\ |O, \lambda)$$

By including the observation sequence in the probability:

$$\gamma_t(j) = \frac{P(q_t = j\ , O|\ \lambda)}{P(O|\lambda)} = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)} \quad ,$$

where by $\alpha_t(j)$ was denoted the forward probability and by $\beta_t(j)$ the backward probability.

So, knowing that for the numerator the sum of $\gamma_t(j)$ for all time steps when the observation o_t was the symbol v_k, and for the denominator the sum of $\gamma_t(j)$ for all time steps *t* should be computed, the expression for $\hat{b}_j(v_k)$ becomes:

$$\hat{b}_j(v_k) = \frac{\sum_{t=1\ s.t.o_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Now, the transition *A* and observation *B* probabilities from an observation sequence *O* can be re-estimated, assuming that at the beginning there are already some previous estimates for *A* and *B*, which represent the initial estimate of the HMM parameters for the forward-backward algorithm. Then, the steps are run iteratively. There are two major steps:

- expectation step, where the expected state occupancy count γ and the expected state transition count ξ are computed from the old *A* and *B* probabilities
- maximization step, where the new *A* and *B* probabilities are computed from γ and ξ

4.4 HMM APPLIED TO SPEECH

The principles when using HMM for speech recognition are the same as the ones from the examples given before, but for the observation sequence, which in this case is a sequence of acoustic feature vectors. Each one of these acoustic feature vectors gives information about the amount of energy in different frequency bands at each point in the time domain. Each observation contains, in fact, a vector of 39 real-valued features that give information about the spectrum of the signal. [11]

When choosing the hidden states of the hidden Markov model, the number of words used for the database creation is extremely important. For small tasks, that is, a small number of words, the

hidden states correspond to entire words. If the number of words increases, the hidden states in the HMM correspond to smaller units, that are called phones. The phone is defined as "the smallest identifiable unit found in a stream of speech that is able to be transcribed" [14]. This way, for larger tasks, the words are considered a sequence of phones, so a word HMM consists of a stream of HMM states. [11]

A major aspect that differentiates the HMM models for speech recognition from other HMM models is the forbiddance of arbitrary transitions. Strong constraints on transitions are imposed, based on the sequential nature of speech. So, states can transition to themselves or to the next state only. This special HMM structure is named Bakis network and is the most common model used for speech.

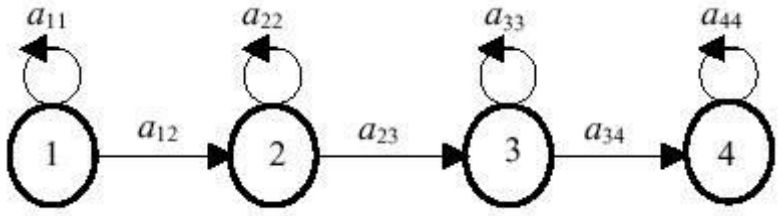

Figure 4.4.1 Bakis Model [15]

The phone durations vary, so self-loops are used in order to allow the repetition of a single phone such that it covers a variable amount of the acoustic input. More than that, the spectral characteristics of a phone and the amount of energy vary across the phone. This is the reason why, in general, a phone is modeled using more than one HMM state. The most common configuration is using three HMM states: beginning, middle and end state.

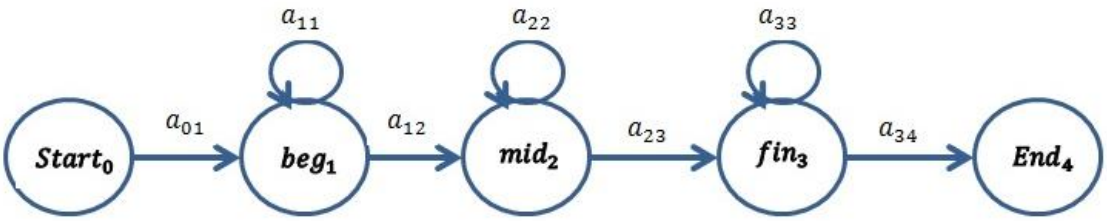

Figure 4.4.2 Standard five-state HMM model for a phone [11]

So, to construct the HMM for a whole word, each phone is replaced by this more complex representation, as represented in the following figure:

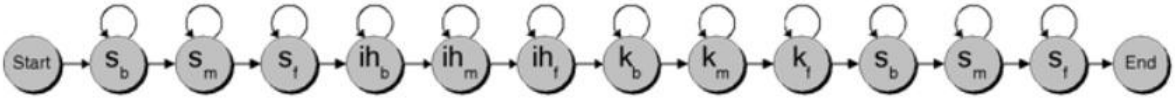

Figure 4.4.3 Composite Model for Word "six" [16]

4.5 MEL FREQUENCY CEPSTRAL COEFFICIENTS (MFCC) VECTORS

The most common feature extracted from the input waveform in speech processing applications is the MFCC. The first step in processing speech is represented by the analog-to-digital conversion, which in turn has two steps itself: sampling and quantization. Sampling means taking the amplitude at certain intervals of time. The sampling rate is the number of samples taken in a second. To be able to accurately measure a waveform, two samples per cycle should be taken, one for the positive part and one for the negative part of the wave. The more samples per cycle taken, the better accuracy. So, the maximum frequency that can be measured is half of the sampling rate and it is called the Nyquist frequency. Most information in human speech is contained in frequencies below 10 kHz, but, for example in telephony, only frequencies below 4 kHz are transmitted, the speech being filtered by the switching network. [11]

Quantization is the process of representing real-valued numbers as integers. All the values that are smaller than the quantum size are represented identically, which means some granular noise appears.

The following diagram represents the steps taken in order to compute the feature vectors:

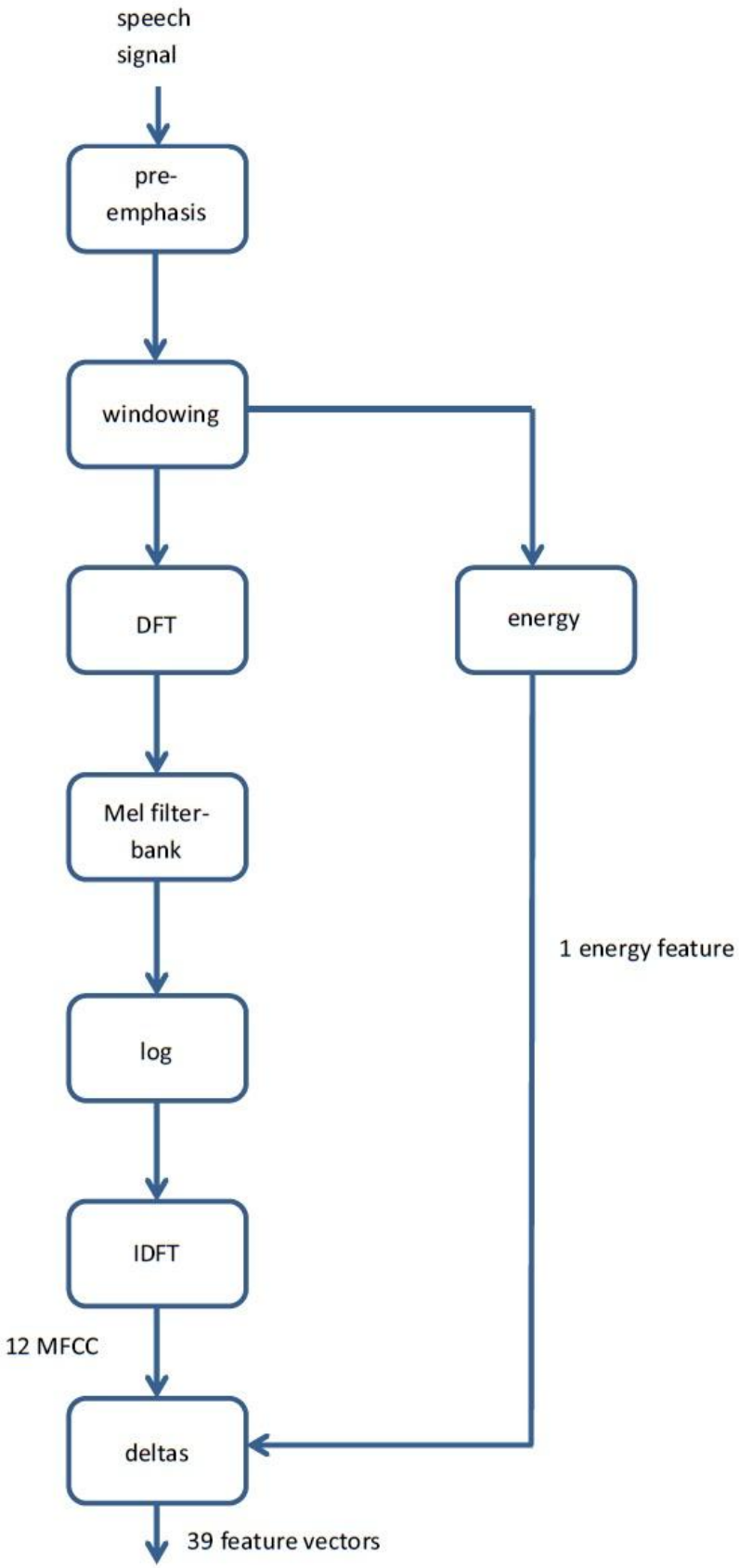

Figure 4.5.1 Extracting MFCC

4.5.1 *Pre-emphasis*

Pre-emphasis is the first step in MFCC feature extraction. It is done due to the fact that it can be observed that the energy for vowels is concentrated on low frequencies and at high frequencies it drops. So, by boosting the high frequency energy, the phone detection accuracy is increased. [11]

4.5.2 *Windowing*

Because the spectral characteristics of the voice signal are not constant in time, it is said that the speech is a non-stationary signal. To be able to extract spectral features, a stationary signal is required. A stationary portion of speech is extracted by using a window which is non-zero inside some region and zero elsewhere.

A windowing process is characterized by:

- the width of the window, in milliseconds
- the offset between successive windows
- the shape of the window

The speech extracted from each window is called a frame, the frame size is represented by the number of milliseconds in a frame and the frame shift is the number of milliseconds between the left edges of two successive frames. [11]

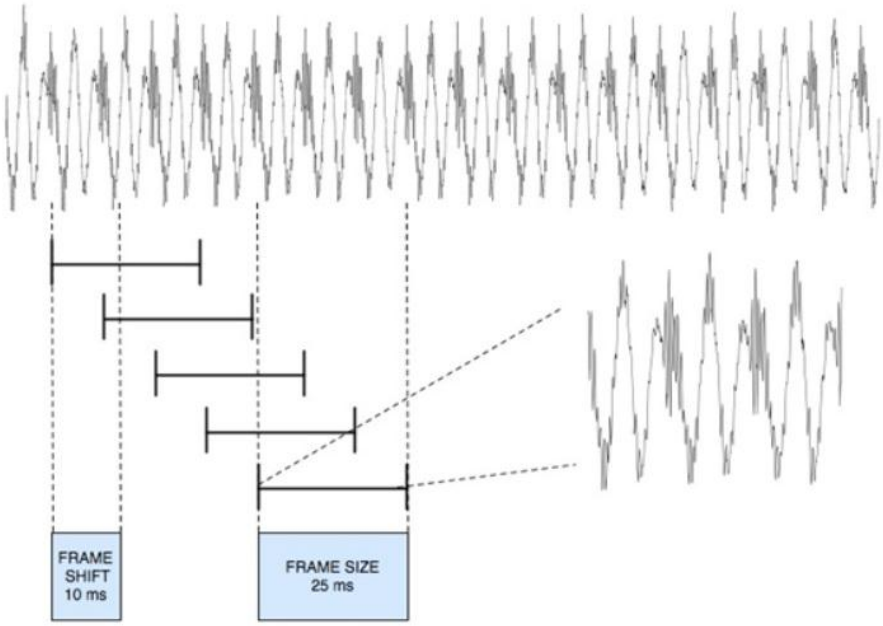

Figure 4.5.2.1 Windowing Process for Frame Shift of 10ms and Frame Size of 25ms [11]

To extract the signal *y[n]*, the value of the input signal, *s[n]*, is multiplied by the value of the window, *w[n]*:

$$y[n] = s[n]w[n]$$

The simplest window is the rectangular window, defined as:

$$w[n] = \begin{cases} 1 & ,0 \leq n \leq L-1 \\ 0 & , \quad otherwise \end{cases}$$

However, problems occur when using the rectangular window due to the abrupt slope that causes discontinuities, thus creating problems when computing the Fourier transform. This inconvenience led to the usage of the Hamming window when extracting the MFCC. It diminishes the values of the signal towards zero at the window boundaries, and at the same time avoids the appearance of discontinuities. The Hamming window is defined:

$$w[n] = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{L}\right) & ,0 \leq n \leq L-1 \\ 0 & , \quad otherwise \end{cases}$$

The following figure best underlies the differences when applying the two window shapes to a sinusoidal input signal:

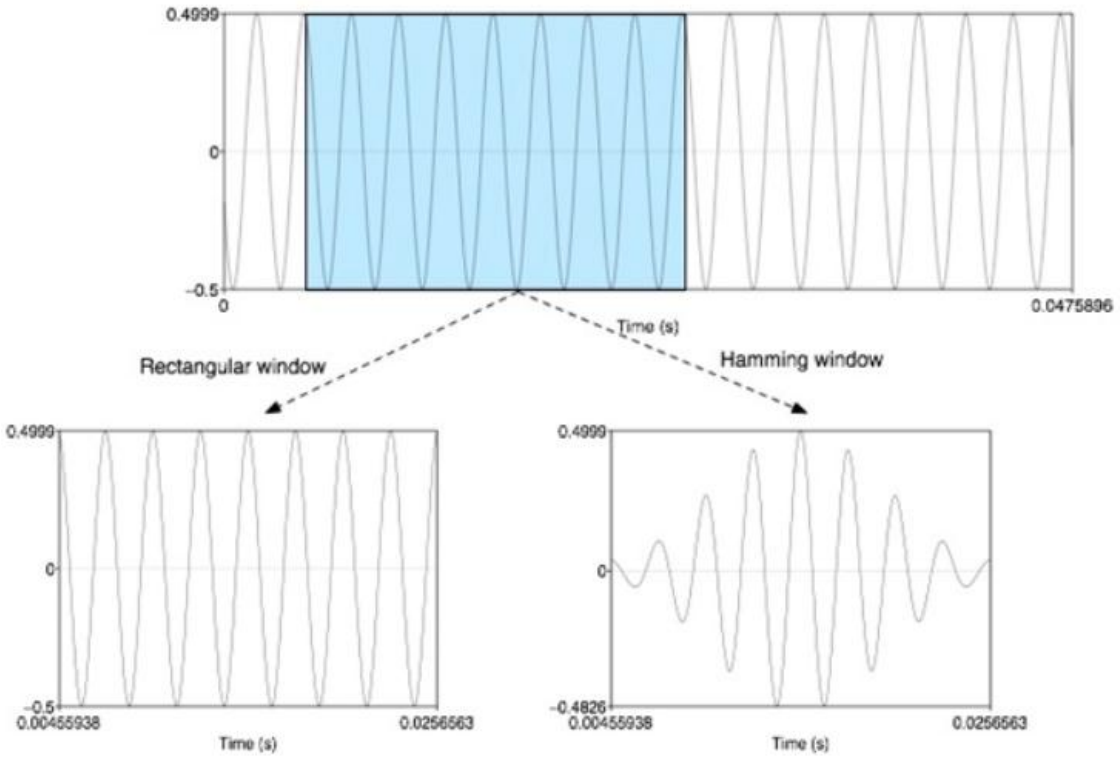

Figure 4.5.2.2 Windowing of sine Wave using Rectangular and Hamming [17]

4.5.3 *Discrete Fourier Transform (DFT)*

The discrete Fourier transformed is used to extract spectral information for discrete frequency bands for sampled signals. At the input, the windowed signal is applied and at the output, for each discrete frequency band, is a complex number, *X[k]*, that represents the magnitude and phase of the frequency component in the original signal. When plotting the magnitude against the frequency, the spectrum of the signal is represented.

The DFT is defined as:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}$$

where N is the number of discrete frequency bands and x[n] are the samples of the input signal.

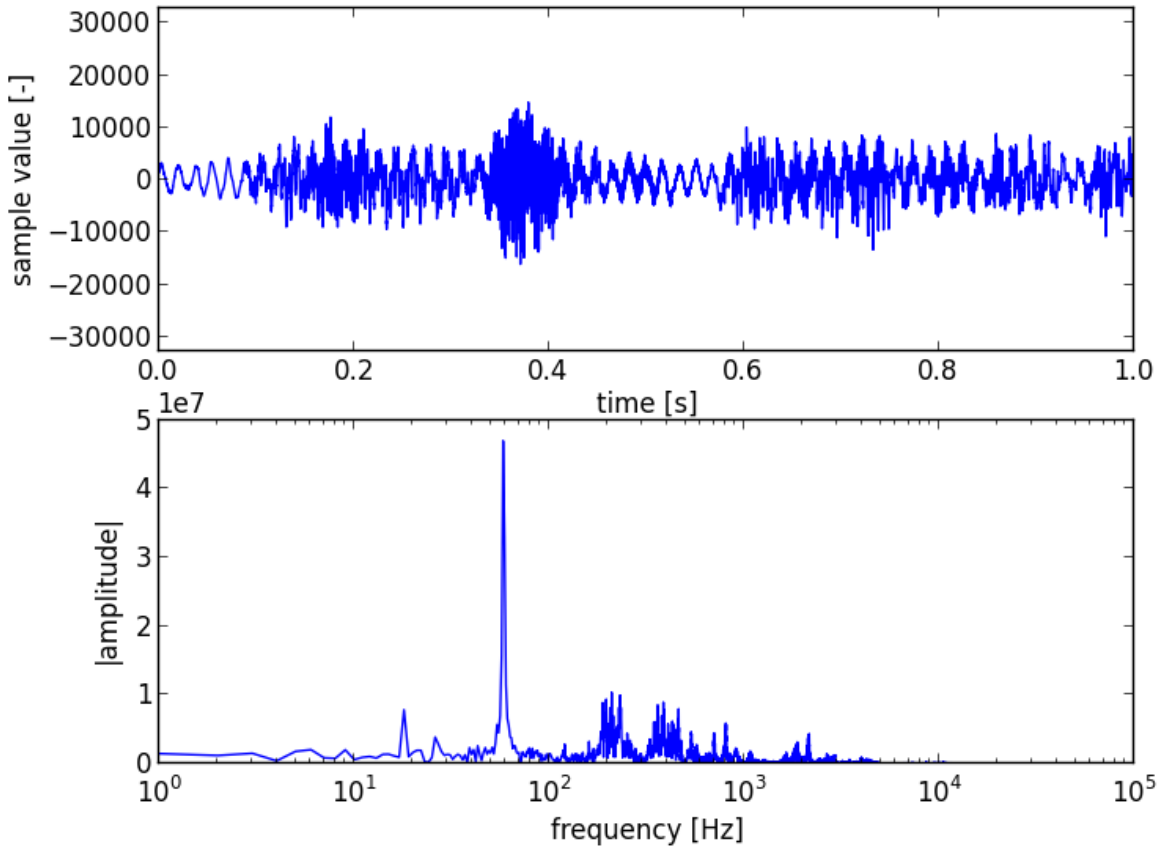

Figure 4.5.3.1 Voice Signal for a 1s Window and its DFT [18]

4.5.4 *Mel Filter Bank and Log*

Since human ear is less sensitive to higher frequency and thus the hearing can be roughly considered logarithmic, a translation of the DFT is needed, such that to increase the accuracy of the results. The translation considered was the wrapping of the frequencies output by DFT onto the mel scale, which is the unit for pitch. An important property is that the sounds that are perceived as equidistant in pitch are separated by an equal number of mels. The mapping between frequency and mel is linear below 1000 Hz and logarithmic above 1000 Hz, as suggested by the transform:

$$mel(f) = 1127 \ln(1 + \frac{f}{700})$$

This mapping is implemented using some triangular filters that collect the energy from each frequency band. Below the frequency of 1000 Hz there are typically 10 filters spread linearly and all other filters above 1000 Hz are spread logarithmically. [11]

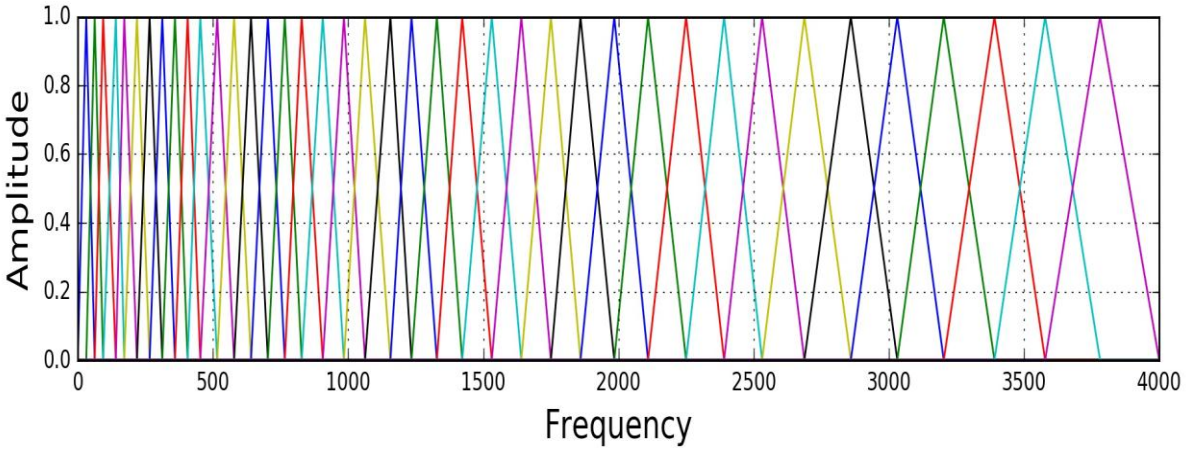

Figure 4.5.4.1 Mel Filter Bank [19]

Then, the log of the mel spectrum values is taken. This is a result of the human response to the signal level being logarithmic, that is, the differences in amplitude at high frequencies bother less than those at low frequencies. More than that, feature estimation thus becomes less sensitive to variations in input, which leads to a better accuracy of results.

4.5.5 *The Inverse Discrete Fourier Transform*

To significantly improve the phone recognition, the cepstrum should be computed. The cepstrum can be thought of as the spectrum of the log of the spectrum. To obtain it, first the standard magnitude spectrum is computed and the log of each amplitude value is taken. Then, this log spectrum is seen as if itself were a waveform. By taking the spectrum of the log spectrum, the representation will be in time domain, so the correct unit for the cepstrum is the sample. When analyzing the cepstrum, it can be observed that a notable component is on a frequency corresponding to the fundamental frequency of the speaker. More than that, at low frequencies, non-zero components appear due to the position of the

tongue and other articulators. So, to detect phones only the low frequency components are needed and to detect the pitch the higher cepstral values are required.

It was observed that the cepstral coefficients have the property that the variance of different coefficients is uncorrelated in general. This is the main reason for working with cepstral coefficients instead directly on the spectrum, where the spectral coefficients at different frequency bands are correlated.

Formally, the cepstrum is defined as the inverse discrete Fourier transform of the log magnitude of the discrete Fourier transform of the signal, so, for a windowed signal *x[n]*, the following expression is employed:

$$c[n] = \sum_{n=0}^{N-1} \log\left(\left|\sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn}\right|\right) e^{j\frac{2\pi}{N}kn}$$

4.5.6 *Deltas and Energy*

The cepstral coefficients are obtained for each frame and typically only the first 12 of them are kept. Next, some other features are added. The energy from the frame correlates with the phone identity so it is an useful feature in phone detection. The energy in a frame is computed as the sum over time of the power of the samples in the frame. So, for a signal *x*, in a window starting from sample t_1 and ending at sample t_2, the energy is computed as: [11]

$$Energy = \sum_{t=t_1}^{t_2} x^2[t]$$

The speech signal differs from frame to frame, so the nature of the change from a stop closure to a stop burst may provide some supplementary information regarding the pitch identity. In order to obtain this new information, for each feature previously discussed (cepstral coefficients and energy) a delta and a double delta feature is added. Delta or velocity features represent the change between frames in the corresponding cepstral or energy feature. Double delta or acceleration features represent the change between frames in the corresponding delta features. [11]

So, after adding all new features, the MFCC features are obtained. The most useful characteristic of the MFCC features is that the cepstral coefficients tend to be uncorrelated, so a simplification of the acoustic model occurs.

4.6 GAUSSIAN MIXTURE MODEL (GMM)

Modern speech recognition algorithms are based on computing observation probabilities directly on the real-valued, continuous input vector. These models are based on the computation of the probability density function (PDF) over a continuous space, the most common model being GMM PDFs. [11]

4.6.1 *Univariate Gaussians*

The Gaussian distribution, also known as the normal distribution, is bell-curved and it is a function parameterized by a mean and a variance. The mean represents the average value and the variance shows the average spread from the mean. The mean is denoted by μ and the variance by σ^2. So, the Gaussian function is:

$$f(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

The mean of a random variable X is the expected value of X. If X is a discrete variable:

$$\mu = E(X) = \sum_{i=1}^{N} p(X_i)\ X_i$$

The variance of a discrete random variable X is the weighted squared average deviation from the mean value:

$$\sigma^2 = E(X_i - E(X))^2 = \sum_{i=1}^{N} p(X_i)(X_i - E(X))^2$$

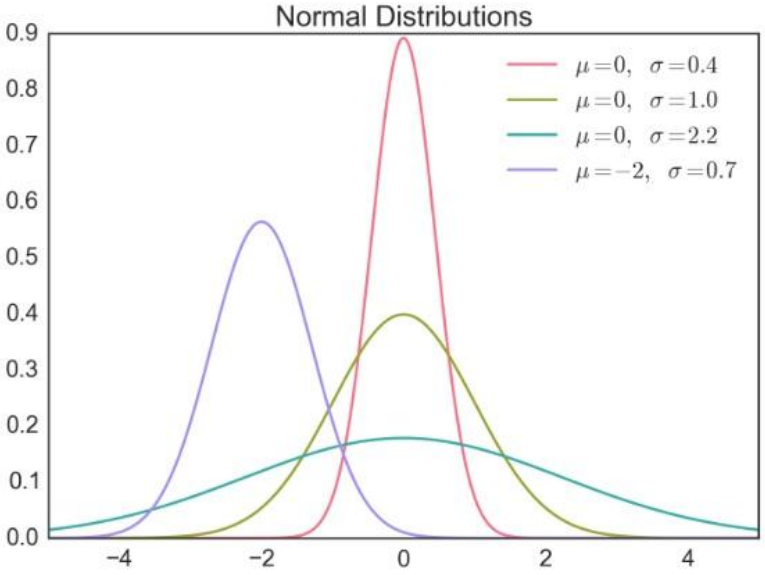

Figure 4.6.1.1 Gaussian Functions with Different Means and Variances [20]

To consider the Gaussian function a PDF, the area under the curve should be 1, so normalization is needed. The probability that a random variable takes values in any interval is the area under the curve between the interval's limits.

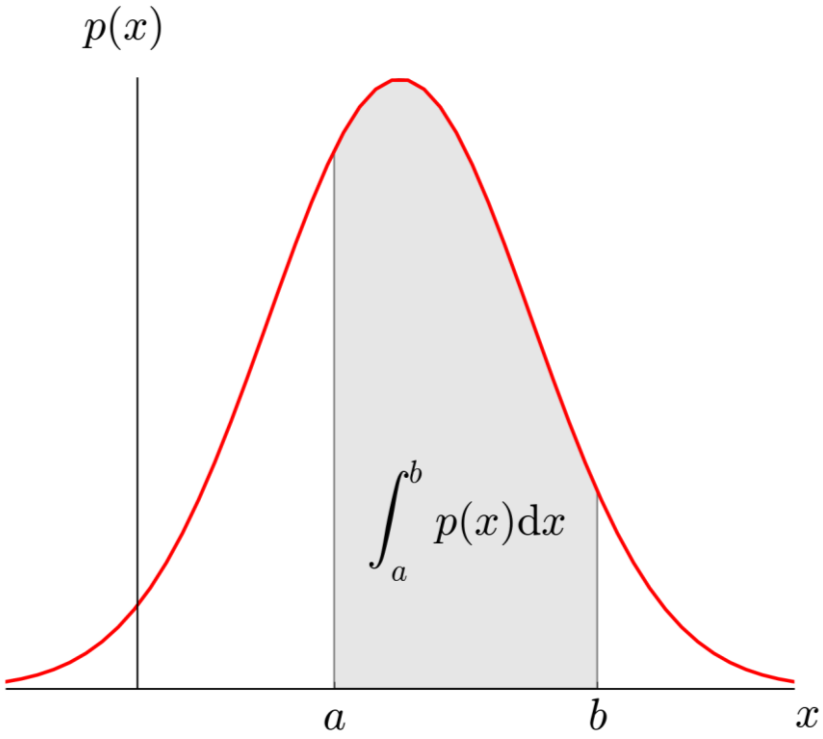

Figure 4.6.1.2 Gaussian PDF [20]

The univariate Gaussian PDF can be used to estimate the probability that a particular HMM state generates the value of a single dimension of a feature vector if the possible values of the observation vector o_t are normally distributed [11]. The observation likelihood function is represented as a Gaussian for one dimension of the acoustic vector. Considering a single cepstral feature and that the state j has a mean value and a variance associated to it, the likelihood is computed using the expression for the Gaussian PDF:

$$b_j(o_t) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{(o_t - \mu_j)^2}{2\sigma_j^2}\right)$$

First, the mean and variance for each HMM state, q_i, should be computed. In the simpler case, when each acoustic observation was labeled with the HMM state that produced it, the mean of each state is computed as the average of the values for each observation vector that correspond to state i. The variance is computed using the mean; it is the sum-squared error between the observation and the mean. [11]

$$\hat{\mu}_i = \frac{1}{T}\sum_{t=1}^{T} o_t \quad s.t.\ q_i\ is\ in\ state\ i$$

$$\hat{\sigma}_j^2 = \frac{1}{T}\sum_{t=1}^{T} (o_t - \mu_i)^2 \quad s.t.\ q_i\ is\ in\ state\ i$$

In reality, because the states are hidden in HMM, it is impossible to know exactly which observation vector was produced by which state. So, each observation vector is assigned to every possible state, given that the HMM was in state *i* at time *t*. The probability of being in state *i* at time *t* was already presented as part of the Baum-Welch algorithm and it was denoted by $\xi_t(i)$. As previously discussed, Baum-Welch is an iterative algorithm, so $\xi_t(i)$ is computed also iteratively because, by getting a better observation probability *b*, a better probability of ξ being in a state *i* at a given time is obtained. Taking all these into account, the expressions for the mean and variance become:

$$\hat{\mu}_i = \frac{\sum_{t=1}^{T} \xi_t(i)\, o_t}{\sum_{t=1}^{T} \xi_t(i)}$$

$$\hat{\sigma}_i^2 = \frac{\sum_{t=1}^{T} \xi_t(i)\, (o_t - \mu_i)^2}{\sum_{t=1}^{T} \xi_t(i)}$$

These two expressions are used in the Baum-Welch algorithm, to train the HMM. Initially, the values for μ_i and σ_i are set to some estimates and then recomputed at each step. [11]

4.6.2 *Multivariate Gaussians*

The use of a multivariate Gaussian is necessary, due to the fact that the acoustic observation is a vector of 39 features. The multivariate Gaussian allows the assignment of a probability to a vector. The multivariate Gaussian is defined by a mean vector $\vec{\mu}$ of *D* elements, *D* being the number of features, and a covariance matrix $\sum$:

$$f(\vec{x}\,|\vec{\mu}, \sum) = \frac{1}{(2\pi)^{\frac{D}{2}}\, |\sum|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\, (x-\mu)^T\, \sum^{-1}(x-\mu)\right)$$

The covariance matrix $\sum$ contains the variance of all dimensions and the covariance between any two dimensions.

$$\sum = \mathrm{E}\big[\big(\mathrm{X} - \mathrm{E(X)}\big)\big(\mathrm{Y} - \mathrm{E(Y)}\big)\big] = \sum_{i=1}^{N} p(X_i Y_i)(X_i - E(X))(Y_i - E(Y))$$

So, the multivariate Gaussian probability estimate for a given HMM, characterized by μ_j and $\sum_j$, is:

$$b_j(o_t) = \frac{1}{(2\pi)^{\frac{D}{2}}\, |\sum|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\, \left(o_t - \mu_j\right)^T\, \sum^{-1}(o_t - \mu_j)\right)$$

The covariance matrix shows the variance between each pair of feature dimensions. If the features in different dimensions are uncorrelated, $\sum_j$ becomes a diagonal matrix, so non-zero elements are placed along the main diagonal. [11]

The following graphical representation best expresses the role of the full covariance matrix, as compared to the one of the diagonal covariance matrix. To simplify the results, the representation was done for a multivariate Gaussian with only two dimensions. The figure from the left depicts the Gaussian with a diagonal covariance matrix, having the variances of the two dimensions equal. The projection of the tridimensional representation is below it and, as it can be seen, the slice is circular. [11]

The figure from the right shows a Gaussian with a non-diagonal covariance matrix. The contour is not lined up with the two axis, so, if the value in one direction is known, the value in the other direction can be predicted. This results in the fact that, having a non-diagonal covariance matrix is equivalent with having correlations between the values of the features in multiple dimensions.

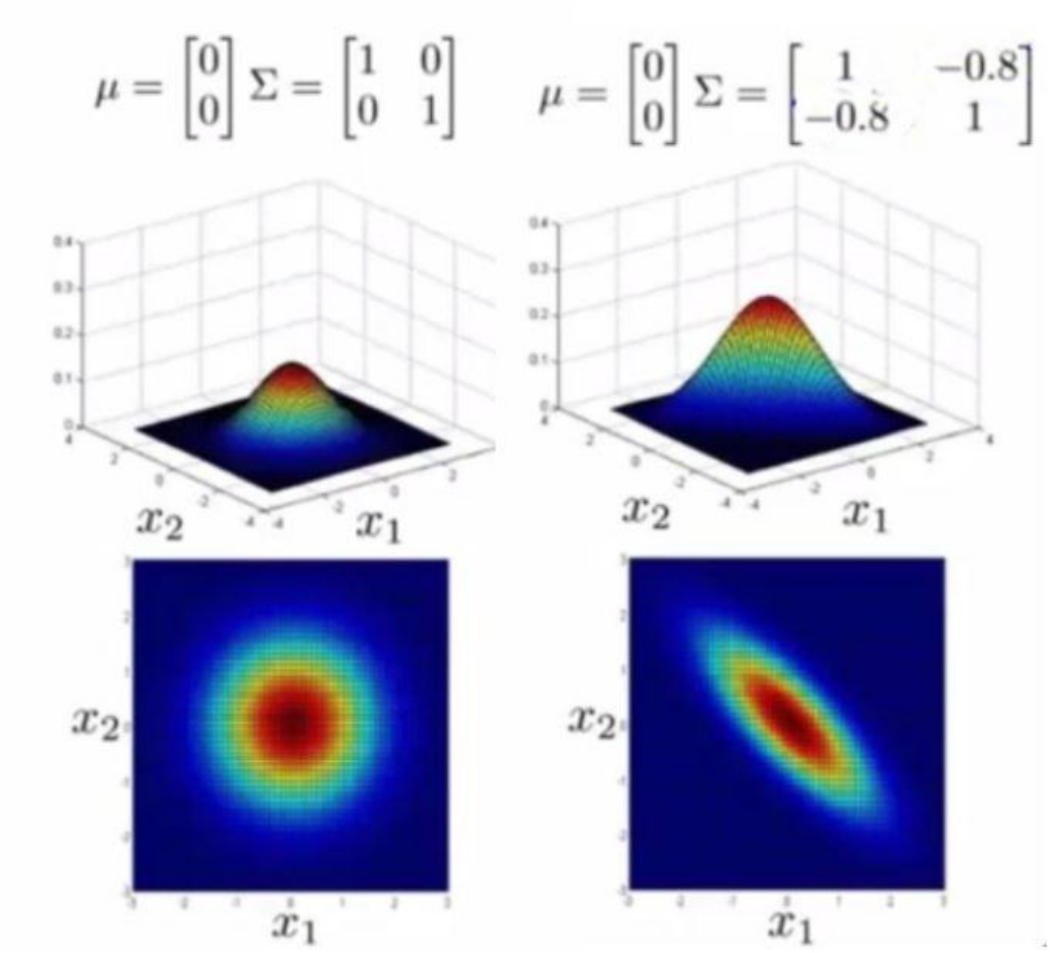

Figure 4.6.2.1 Multivariate Gaussians in Two Dimensions [21]

As previously proven, a Gaussian with a full covariance matrix is a more powerful tool than the diagonal one. But, due to the fact that the full covariance matrix is slow to compute (a full covariance matrix has D^2 parameters) and it has more parameters and thus requires more data to train, the diagonal covariance matrix is generally used for Automatic Speech Recognition (ASR) systems. [11]

So, with this assumption, the covariance matrix becomes:

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_D^2 \end{bmatrix}$$

And, introducing in the expression for the likelihood, the following simplification is obtained:

$$b_j(o_t) = \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\sigma_{jd}^2}} \, exp\left(-\frac{1}{2}\frac{(o_t - \mu_{jd})^2}{\sigma_{jd}^2}\right)$$

To train such a multivariate Gaussian, the same steps as for the univariate Gaussian are taken. Again, the Baum-Welch algorithm is used, having in mind that $\xi_t(i)$ is the likelihood of being in state *i* at time *t*. The same equations are employed, except that now the observation o_t is a vector of cepstral features, the mean vector $\vec{\mu}$ is a vector of cepstral means and the variance vector $\vec{\sigma}_i^2$ is a vector of cepstral variances:

$$\hat{\mu}_i = \frac{\sum_{t=1}^{T} \xi_t(i) o_t}{\sum_{t=1}^{T} \xi_t(i)}$$

$$\hat{\sigma}_i^2 = \frac{\sum_{t=1}^{T} \xi_t(i)(o_t - \mu_i)\,(o_t - \mu_i)^T}{\sum_{t=1}^{T} \xi_t(i)}$$

4.6.3 *Gaussian Mixture Models - Motivation*

As it was previously shown, a multivariate Gaussian can be used to model each dimension of the feature vector as a normal distribution. Generally, a cepstral feature is not normally distributed; for this reason, the observation likelihood is modeled with a weighted mixture of multivariate Gaussians. Such a model is called a Gaussian mixture model. [11] GMM has become the standard classifier for text-independent speaker recognition due to its ability to form smooth approximations to arbitrary shaped distributions. Another advantage is that the training is fast as compared to other methods. [22]

Since HMM is hard to be applied to text-independent speaker recognition, and even so the improvement is not significant, GMM became the most used model. [23]

In GMM, the speaker model consists of a finite mixture of multivariate Gaussian components. A mixture density is the weighted sum of *M* component densities. [24]

$$p(\vec{x}\,|\lambda\,) = \sum_{i=1}^{M} p_i b_i(\vec{x}) \quad ,$$

where $\vec{x}$ is a D dimensional random vector, p_i are the mixture weights and $b_i(\vec{x})$ are the component densities.

Each component density is:

$$b_i(\vec{x}) = \frac{1}{(2\pi)^{\frac{D}{2}} |\sum_i|^{\frac{1}{2}}} exp\left(-\frac{1}{2} (\vec{x} - \vec{\mu}_i)^T \sum_i^{-1} (\vec{x} - \vec{\mu}_i)\right) \quad ,$$

where $\vec{\mu}_i$ is the mean vector and $\sum_i$ is the covariance matrix.

The constraint imposed on the mixture weights is:

$$\sum_{i=1}^{M} p_i = 1$$

The figure below shows how the approximation of an arbitrary function is done, using a mixture of three Gaussians.

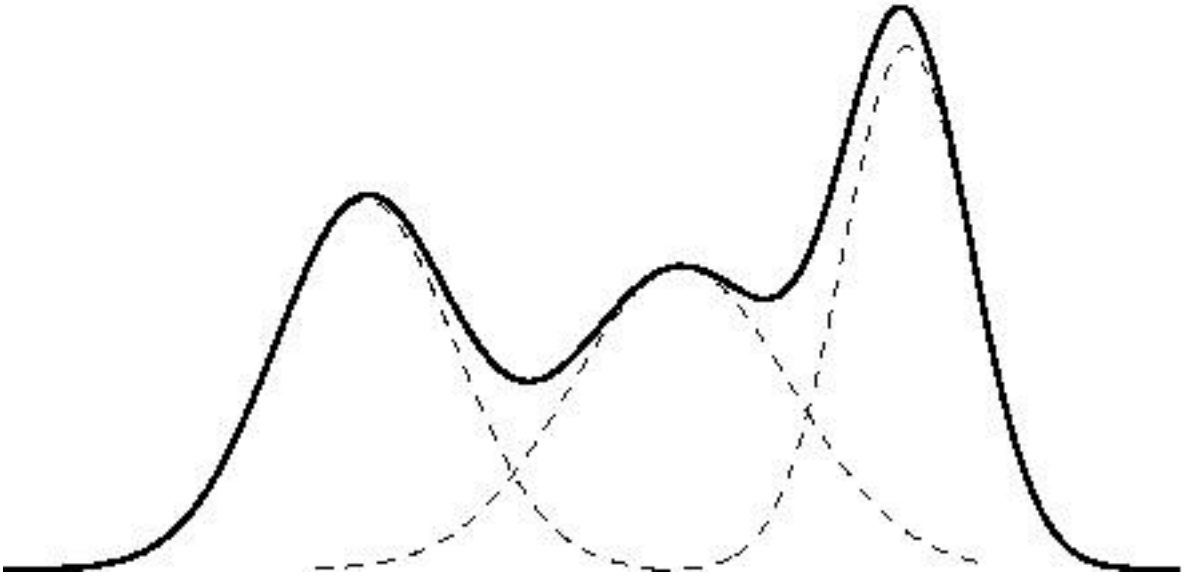

Figure 4.6.3.1 Function Approximation using a Mixture of Three Gaussians [25]

The mixture density is characterized by the parameter λ, which contains the mean vector, covariance matrices and mixture weights:

$$\lambda = \{p_i, \vec{\mu}_i, \sum_i\}, \quad i = 1 \dots M$$

Each speaker is represented by his own GMM and is referred to by his model λ. [24]

To train the GMM, Baum-Welch is used to determine the probability of a certain mixture of Gaussians, being given the observation. This probability will be iteratively updated. The probability of being in state *j* at time *t* with the m^{th} mixture component accounting for the observation o_t is denoted by $\xi_{tm}(j)$ and is defined as: [11]

$$\xi_{tm}(j) = \frac{\sum_{i=1}^{N} \alpha_{t-1}(j) a_{ij} p_{jm} b_{jm}(o_t) \beta_t(j)}{\alpha_T(F)}$$

Using the values from the previous iteration, the mean, mixture weight and covariance are recomputed:

$$\hat{\mu}_{im} = \frac{\sum_{t=1}^{N} \xi_{tm}(i)\, o_t}{\sum_{t=1}^{T} \sum_{m=1}^{M} \xi_{tm}(i)}$$

$$\hat{p}_{im} = \frac{\sum_{t=1}^{N} \xi_{tm}(i)}{\sum_{t=1}^{T} \sum_{m=1}^{M} \xi_{tm}(i)}$$

$$\hat{\Sigma}_{im} = \frac{\sum_{t=1}^{N} \xi_t(i)(o_t - \mu_{im})(o_t - \mu_{im})^T}{\sum_{t=1}^{T} \sum_{m=1}^{M} \xi_{tm}(i)}$$

There are two main reasons why GMM is used in speaker recognition applications. The first one is that GMM may model some underlying set of acoustic classes, which represent some phonetic events, such as vowels, nasals or fricatives. They represent some speaker-dependent features that are useful in speaker identification. The second reason was obtained through repeated observations and as it was concluded, a linear combination of Gaussian basis functions offers the possibility to represent a large class of sampled distributions. Any arbitrarily-shaped function can be well approximated through GMM. [24]

Another important characteristic is that, even though there might be some correlation between the features, full covariance matrices are not needed. Correlations between feature vector elements can be modeled using a linear combination of diagonal covariance Gaussians.

4.6.4 *Maximum Likelihood Parameter Estimation*

The goal of speaker recognition model is to estimate the GMM parameters that best match the distribution of training feature vectors. The most used method to reach this imposed target is the maximum likelihood estimation. Maximum likelihood estimation aims to determine the model parameters that maximize the likelihood of the GMM, when the training data is already given.

Considering *T* training vectors, $X = \{\vec{x}_1, \ldots, \vec{x}_T\}$, the GMM likelihood is:

$$p(X \mid \lambda) = \prod_{t=1}^{T} p(\vec{x}_t \mid \lambda)$$

Seeing that this expression cannot be maximized directly, as it is a nonlinear function of the parameters λ, the maximum likelihood estimation is obtained iteratively, using the particular case of expectation-maximization (EM) algorithm.

EM algorithm starts with an initial model λ and estimates a new model $\bar{\lambda}$ such that the following condition is fulfilled:

$$p(X\,|\bar{\lambda})\ \geq p(X|\lambda)$$

After this step, the newly obtained model becomes the initial one for the next iteration and the process repeats until some threshold is reached. As it can be observed, the principles are the same as in the case of Baum-Welch algorithm, used to estimate HMM parameters. [24]

At each iteration, the weight, mean and variance are updated using:

$$\bar{p_i} = \frac{1}{T}\sum_{t=1}^{T} p(i|\vec{x}_t, \lambda)$$

$$\vec{\bar{\mu}}_i = \frac{\sum_{t=1}^{T} p(i|\vec{x}_t, \lambda)\vec{x}_t}{\sum_{t=1}^{T} p(i|\vec{x}_t, \lambda)}$$

$$\bar{\sigma}_i^2 = \frac{\sum_{t=1}^{T} p(i|\vec{x}_t, \lambda)\ x_t^2}{\sum_{t=1}^{T} p(i|\vec{x}_t, \lambda)} - \bar{\mu}_i^2$$

With σ_i^2 , μ_i and x_t are denoted the elements from the vectors $\vec{\sigma}_i^2$, $\vec{\mu}_i$ and $\vec{x}_t$.

The a posteriori probability for a class *i* is determined using:

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

Two important steps that are taken when training the Gaussian mixture model are the selection of the order M of the mixture and the initialization of parameters for the estimation maximization algorithm. Since there is no theoretical background to impose when choosing these parameters, they are application dependent and the values are selected through repeated experiments. [24]

Usually, the *X* feature vectors are assumed independent, so the logarithm of the conditional probability is computed:

$$\log\left(p\left(\vec{X}\,|\boldsymbol{\lambda}\right)\right) = \sum_{t=1}^{T} \log(p(\overrightarrow{x_t}|\lambda))$$

$p(\overrightarrow{x_t}|\lambda)$ is computed as previously stated,

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

Sometimes, a normalization through the division by *T* is necessary; this can be considered a rough compensation factor to the likelihood value. [26]

4.7 UNIVERSAL BACKGROUND MODEL (UBM)

The UBM is "a model in speaker verification system to represent general, person-independent, channel independent feature characteristics to be compared against a model of speaker specific feature characteristics when making an accept or reject decision". [23]

The UBM acts as a prior model in maximum a posteriori (MAP) parameter estimation and it is trained using samples from many speakers, in order to have some general speech characteristics.

Since there is no possibility to determine the optimal number of speakers or speech samples to be used when training the UBM, the simplest method is to pool all data using EM algorithm. This is done in order to avoid the dominance of one subpopulation over the others.

4.7.1 *Adaptation of Speaker Model*

Unlike the standard approach of maximum likelihood training of a model independently on the UBM, the adaptation procedure is used to continuously update the parameters in the UBM. This method drastically increases the performances, since the speaker's model and the UBM are tightly connected.

The adaptation is performed in two steps. In the first step, for each mixture in the UBM, the estimates of the sufficient statistics of the speaker's training data are computed. The sufficient statistics are the basic statistics needed in order to compute the desired parameters. In the second step, the new results are combined with the old ones using a data-dependent mixing coefficient. This coefficient is chosen such that, when there is enough data the new sufficient statistics are more reliable for final parameter estimation, and when the data count is low, the final result relies more on the a priori information. [26]

Being given an UBM and the training vectors $X = \{x_1, \dots, x_T\}$, the probabilistic alignment of the training vectors in the UBM mixture is first determined. For mixture *i*, the probabilistic alignment is:

$$\Pr(i|\vec{x}_t) = \frac{p_i b_i(\vec{x}_t)}{\sum_{j=1}^{M} p_j b_j(\vec{x}_t)}$$

Then, this probabilistic alignment is used to compute the sufficient statistics for weight, mean and variance:

$$n_i = \sum_{t=1}^{T} \Pr(i|\vec{x}_t)$$

$$E_i(\vec{x}) = \frac{1}{n_i} \sum_{t=1}^{T} \Pr(i|\vec{x}_t)\, \vec{x}_t$$

$$E_i(\vec{x}^2) = \frac{1}{n_i} \sum_{t=1}^{T} \Pr(i|\vec{x}_t)\ \vec{x}_t^2$$

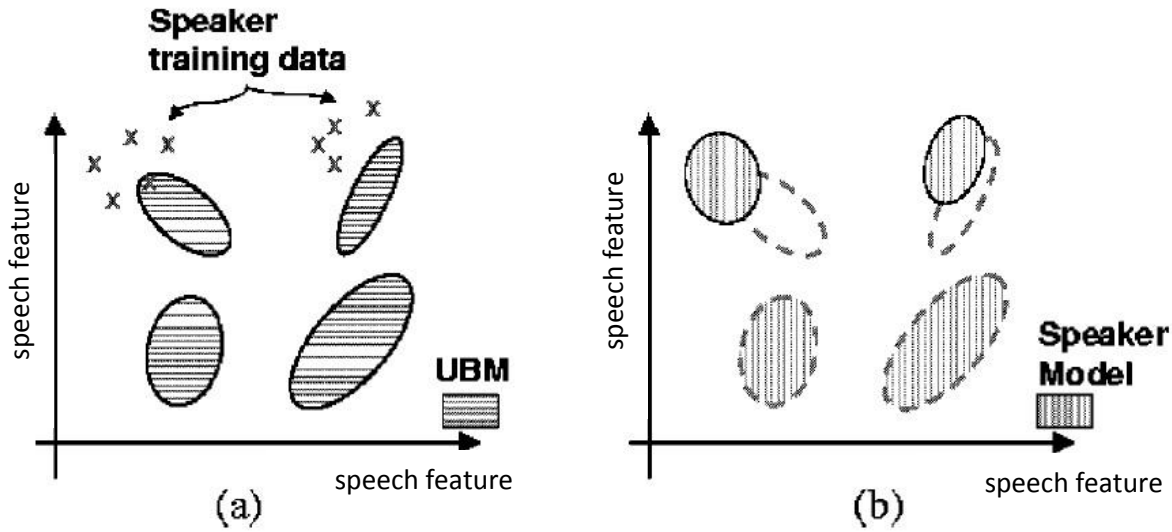

Figure 4.7.1.1 Adaptation for a Speaker Model [25]

Figure a) shows how the training vectors are mapped into the UBM mixture. Figure b) represents the adapted mixture parameters that are derived using the UBM mixture parameters and the statistics of the new data. [25]

After this step, the sufficient statistics from the training data are used to update the old sufficient statistics form de UBM for mixture *i*, thus resulting the adapted parameters for that specific mixture:

$$\hat{p}_i = \left[\frac{\alpha_i^p n_i}{T} + (1 - \alpha_i^p)p_i\right]\gamma$$

$$\hat{\mu}_i = \alpha_i^m E_i(\vec{x}) + (1 - \alpha_i^m)\mu_i$$

$$\hat{\sigma}_i^2 = \alpha_i^v E_i(\overrightarrow{x^2}) + (1 - \alpha_i^v)(\sigma_i^2 + \mu_i^2) - \hat{\mu}_i^2$$

$\{\alpha_i^p, \alpha_i^m, \alpha_i^v\}$ are the adaptation parameters for the weight, mean and variance and γ is the scale factor, computed over all adaptation mixture weights, such that to impose the condition that their sum is 1. [26]

The adaptation parameter coefficient that is computed for each parameter and each mixture, is defined as follows:

$$\alpha_i^\rho = \frac{n_i}{n_i + r^\rho}$$

where $\rho\epsilon\{p, m, v\}$ and $r^\rho$is a fixed relevance factor for parameter ρ.

If the probabilistic count, n_i, is low for a specific mixture component, it will make $\alpha_i^\rho \rightarrow 0$. This, in turn, leads to a de-emphasis of the new parameters, thus giving the old parameters more weight. For mixture components with high probabilistic count, $\alpha_i^\rho \rightarrow 1$, so the new speaker-parameters are prioritized over the a priori parameters. Through the relevance factor the amount of new data to be observed in a mixture before the replacing of old parameters with the new ones is controlled.

The experiments carried out by Vuuren in his Ph.D. thesis, "Speaker Verification in a Time-Feature Space" proved that the gain when using parameter-dependent adaptation coefficients is insignificant. [27] So, in most GMM-UBM system, a single adaptation coefficient for all parameters is used, having the relevance factor 16. Vuuren's experiments show similar performances for relevance factors in the range [8-20].

The adaptation approach provides by far better results as compared to the method in which the speaker model is trained independently on the UBM. If the UBM is considered the covering space of speaker-independent acoustic classes, the adaptation is "the speaker-dependent tuning of those acoustic classes observed in the speaker's training speech". [26] During the recognition stage, the classes unseen in the speaker training produce zero log-likelihood ratio values, that are this way not taken into account in the decision making process.

4.7.2 *Log-Likelihood Ratio Computation*

For a sequence of feature vectors *X*, the log-likelihood ratio is computed as:

$$\Lambda(X) = \log p\left(\vec{X}\middle|\lambda_{hyp}\right) - \log p(\vec{X}|\lambda_{UBM})$$

Since the hypothesized speaker model was adapted from the UBM, the method works faster than evaluating separately two GMMs. This is due to the fact that, when evaluating a large GMM for a feature vector, only a few of the mixtures yield significantly to the likelihood value. The GMM is represented over a large space, but only few components are near a single vector. [26]

Another observation that was made is that the vectors that are close to a particular mixture in the UBM are also close to the corresponding mixture in the speaker model. So, having these two properties in mind, an improvement in the latency of the response is obtained. [26]

Following all the steps described in this chapter, the diagram below represents the method implemented in order to obtain the desired result:

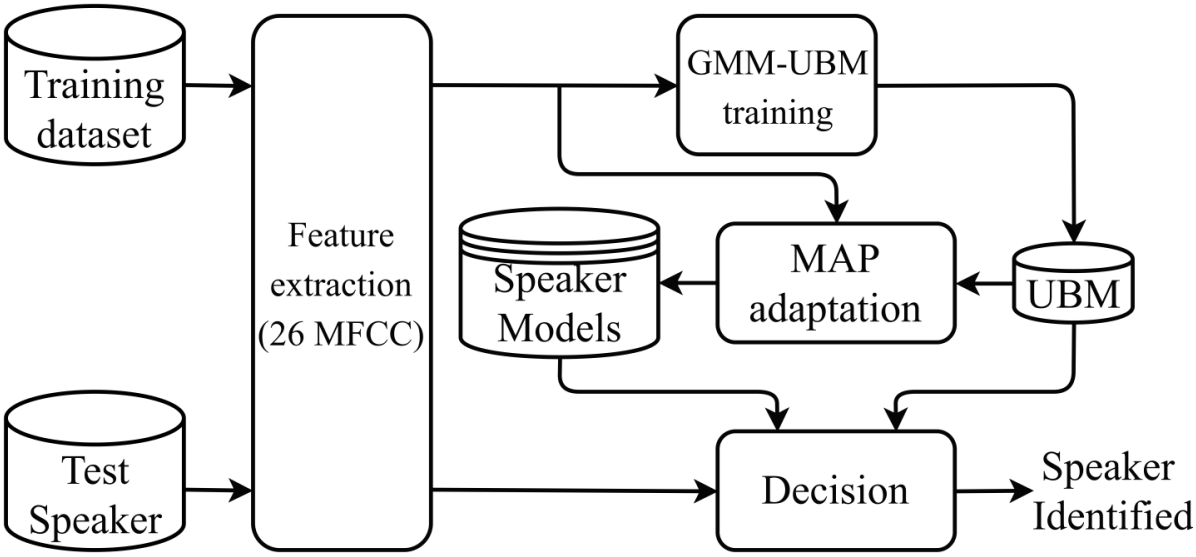

Figure 4.1 Implemented Method

CHAPTER 5
TESTING THE METHOD

5.1 DATABASE

To create the database used to test the algorithm, a set of 5 speakers was used. For each speaker, 20 voice commands were recorded 10 times each, from different distances and positions relative to the robot's microphones, in order to have a more accurate model of the speech for the users. The algorithm developed is text dependent, meaning that the same commands that were used for training the model should be used in the testing phase.

The 5 speakers chosen were two males and three females, such that to have some diversity. The recordings were done in a nosy environment, in order to study the effect of noise on the overall accuracy. The 10 recordings for each command were further divided as follows:

- 5 recordings for distances below 1 meter
- 5 recordings for distances greater than 1 meter

Each of these recordings was done from a different position relative to the robot.

The channel is stereo, the recording being done on 2 of the 4 microphones available on the robot, the ones placed on the right and left side of its head. This decision was made after studying the quality of some test recordings from each microphone.

5.2 EXPERIMENTAL SETUP

The database thus obtained was divided into:

- 75% of the recordings were used for training
- 25% of the recordings were used for testing

To make the choice of optimal parameters, it was studied the effect of the sampling rate and number of Gaussian components variation on the overall results. For the sampling frequency, the values for which the study was made are:

- 8 kHz
- 16 kHz
- 32 kHz
- 44.1 kHz

The number of Gaussian components was varied as follows:

- 1 Gaussian component
- 2 Gaussian components
- 4 Gaussian components
- 8 Gaussian components
- 16 Gaussian components
- 32 Gaussian components
- 64 Gaussian components
- 128 Gaussian components
- 256 Gaussian components

5.3 RESULTS

The results of the testing for each variant are given below. First, the results obtained for the noisy environment are presented as follows:

Fs = 8kHz 1 Gaussian Component		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	98	0	0	2
	Speaker 3	0	0	100	0	0
	Speaker 4	1	0	0	88	11
	Speaker 5	0	10	0	17	73

Table 5.3.1 Results for 1 Gaussian component and 8 kHz sampling frequency – noisy environment

Fs = 8kHz 2 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	99	0	1	0
	Speaker 3	0	0	100	0	0
	Speaker 4	2	0	0	90	8
	Speaker 5	0	9	0	13	78

Table 5.3.2 Results for 2 Gaussian components and 8 kHz sampling frequency – noisy environment

Fs = 8kHz 4 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	97	0	0	3	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	4	0	94	2
	Speaker 5	0	11	0	3	86

Table 5.3.3 Results for 4 Gaussian components and 8 kHz sampling frequency – noisy environment

Fs = 8kHz 8 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	98	2
	Speaker 5	0	5	0	2	93

Table 5.3.4 Results for 8 Gaussian components and 8 kHz sampling frequency – noisy environment

Fs = 8kHz 16 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	1	0	1	98

Table 5.3.5 Results for 16 Gaussian components and 8 kHz sampling frequency – noisy environment

Fs = 8kHz 32 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	1	0	2	97

Table 5.3.6 Results for 32 Gaussian components and 8 kHz sampling frequency – noisy environment

Fs = 8kHz 64 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	0	0	0	100

Table 5.3.7 Results for 64 Gaussian components and 8 kHz sampling frequency – noisy environment

Fs = 8kHz 128 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	2	0	0	98

Table 5.3.8 Results for 128 Gaussian components and 8 kHz sampling frequency – noisy environment

Fs = 8kHz 256 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	0	0	0	100

Table 5.3.9 Results for 256 Gaussian components and 8 kHz sampling frequency – noisy environment

Acceptable results are obtained starting with 16 Gaussian components for this sampling frequency. It can be observed that, if 256 components are used, the accuracy is 100%.

Fs = 16kHz 1 Gaussian Component		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	93	7
	Speaker 5	0	5	0	14	81

Table 5.3.10 Results for 1 Gaussian component and 16 kHz sampling frequency – noisy environment

Fs = 16kHz 2 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	93	7
	Speaker 5	0	1	0	7	92

Table 5.3.11 Results for 2 Gaussian components and 16 kHz sampling frequency – noisy environment

Fs = 16kHz 4 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	96	4
	Speaker 5	0	1	0	7	92

Table 5.3.12 Results for 4 Gaussian components and 16 kHz sampling frequency – noisy environment

Fs = 16kHz 8 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	4	0	1	95

Table 5.3.13 Results for 8 Gaussian components and 16 kHz sampling frequency – noisy environment

Fs = 16kHz 16 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	99	1
	Speaker 5	0	1	0	0	99

Table 5.3.14 Results for 16 Gaussian components and 16 kHz sampling frequency – noisy environment

Fs = 16kHz 32 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	2	0	0	98

Table 5.3.15 Results for 32 Gaussian components and 16 kHz sampling frequency – noisy environment

Fs = 16kHz 64 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	0	0	0	100

Table 5.3.16 Results for 64 Gaussian components and 16 kHz sampling frequency – noisy environment

Fs = 16kHz 128 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	0	0	0	100

Table 5.3.17 Results for 128 Gaussian components and 16 kHz sampling frequency – noisy environment

Fs = 16kHz 256 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	0	0	0	100

Table 5.3.18 Results for 256 Gaussian components and 16 kHz sampling frequency – noisy environment

When employing a 16 kHz sampling frequency, the quality of results increases faster. The 8 Gaussian components approximation gives reliable results.

Fs = 32kHz 1 Gaussian Component		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	87	13
	Speaker 5	0	7	0	8	85

Table 5.3.19 Results for 1 Gaussian component and 32 kHz sampling frequency – noisy environment

Fs = 32kHz 2 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	90	10
	Speaker 5	0	1	0	7	92

Table 5.3.20 Results for 2 Gaussian components and 32 kHz sampling frequency – noisy environment

Fs = 32kHz 4 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	98	2
	Speaker 5	0	2	0	5	93

Table 5.3.21 Results for 4 Gaussian components and 32 kHz sampling frequency – noisy environment

Fs = 32kHz 8 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	0	0	1	99

Table 5.3.22 Results for 8 Gaussian components and 32 kHz sampling frequency – noisy environment

Fs = 32kHz 16 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	1	0	0	99

Table 5.3.23 Results for 16 Gaussian components and 32 kHz sampling frequency – noisy environment

Fs = 32kHz 32 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	0	0	0	100

Table 5.3.24 Results for 32 Gaussian components and 32 kHz sampling frequency – noisy environment

Fs = 32kHz 64 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	0	0	0	100

Table 5.3.25 Results for 64 Gaussian components and 32 kHz sampling frequency – noisy environment

Fs = 32kHz 128 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	0	0	0	100

Table 5.3.26 Results for 128 Gaussian components and 32 kHz sampling frequency – noisy environment

Fs = 32kHz 256 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	0	0	0	100

Table 5.3.27 Results for 256 Gaussian components and 32 kHz sampling frequency – noisy environment

For the 32 kHz sampling frequency and 8 Gaussian components, the overall accuracy is 99.8%. If more components are used, it increases to 100%.

Fs = 44.1kHz 1 Gaussian Component		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	92	8
	Speaker 5	0	0	0	12	88

Table 5.3.28 Results for 1 Gaussian component and 44.1 kHz sampling frequency – noisy environment

Fs = 44.1kHz 2 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	96	4
	Speaker 5	0	1	0	9	90

Table 5.3.29 Results for 2 Gaussian components and 44.1 kHz sampling frequency – noisy environment

Fs = 44.1kHz 4 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	94	6
	Speaker 5	0	1	0	4	95

Table 5.3.30 Results for 4 Gaussian components and 44.1 kHz sampling frequency – noisy environment

Fs = 44.1kHz 8 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	99	1
	Speaker 5	0	1	0	0	99

Table 5.3.31 Results for 8 Gaussian component sand 44.1 kHz sampling frequency – noisy environment

Fs = 44.1kHz 16 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	99	1
	Speaker 5	0	2	0	0	98

Table 5.3.32 Results for 16 Gaussian components and 44.1 kHz sampling frequency – noisy environment

Fs = 44.1kHz 32 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	99	1
	Speaker 5	0	0	0	1	99

Table 5.3.33 Results for 32 Gaussian components and 44.1 kHz sampling frequency – noisy environment

Fs = 44.1kHz 64 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	98	2
	Speaker 5	0	0	0	0	100

Table 5.3.34 Results for 64 Gaussian components and 44.1 kHz sampling frequency – noisy environment

Fs = 44.1kHz 128 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	98	2
	Speaker 5	0	0	0	0	100

Table 5.3.35 Results for 128 Gaussian components and 44.1 kHz sampling frequency – noisy environment

Fs = 44.1kHz 256 Gaussian Components		Original				
		Speaker 1	Speaker 2	Speaker 3	Speaker 4	Speaker 5
Predicted	Speaker 1	100	0	0	0	0
	Speaker 2	0	100	0	0	0
	Speaker 3	0	0	100	0	0
	Speaker 4	0	0	0	100	0
	Speaker 5	0	0	0	0	100

Table 5.3.36 Results for 265 Gaussian components and 44.1 kHz sampling frequency – noisy environment

The overall accuracy is defined as:

$$Overall\ Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Samples}$$

In the following figure the overall accuracy is presented. On the *x* axis, number of Gaussians was considered and on the *y* axis the overall accuracy. This graphical representation was done for each of the four sampling frequencies used in this study.

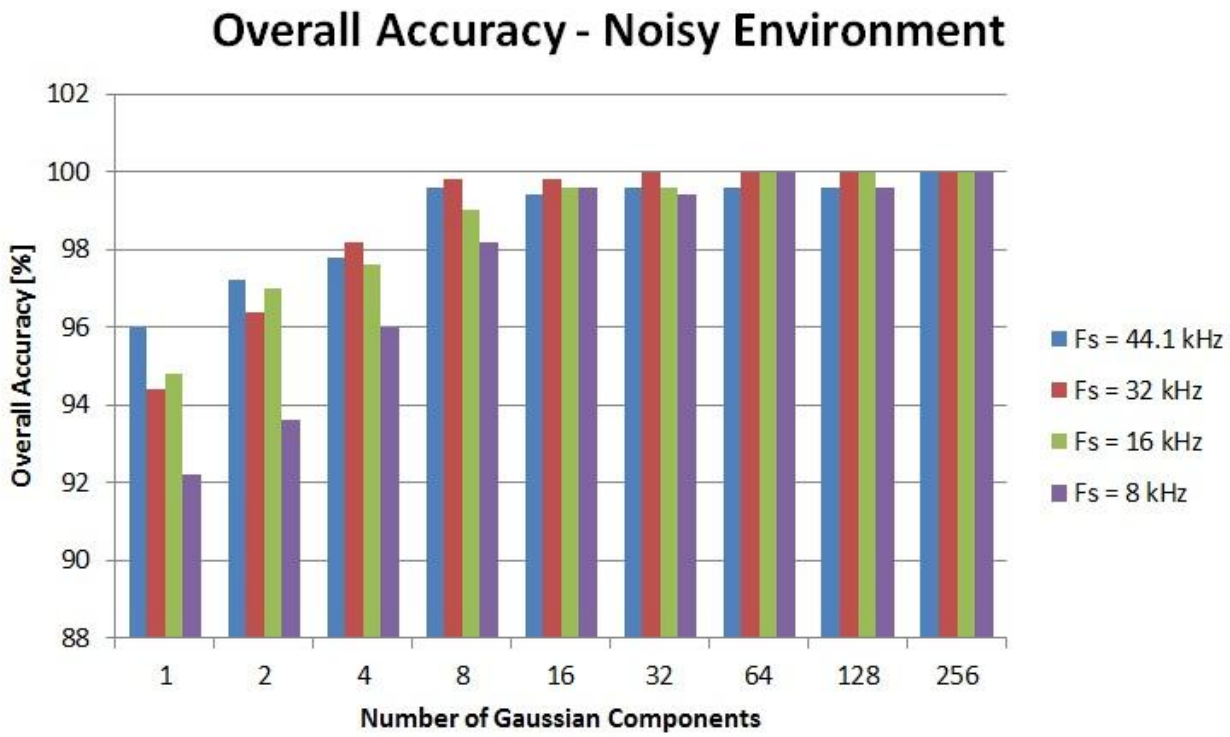

Figure 5.3.1 Overall Accuracy for Noisy Recordings

As suggested by this graphical representation, the accuracy increases with the increase of the sampling frequency and number of Gaussian components used to approximate the model. It can be observed that, starting from 16 Gaussian components, the results are similar, regardless of the sampling frequency. More than that, the worst case is represented by the 8 kHz sampling rate and only 1 Gaussian component; still, the overall accuracy is of 92%.

The recordings were then filtered using Wiener filtering and the model was trained again, using this time the filtered recordings. Similar confusion matrices were obtained, and the overall accuracy graphical representation is:

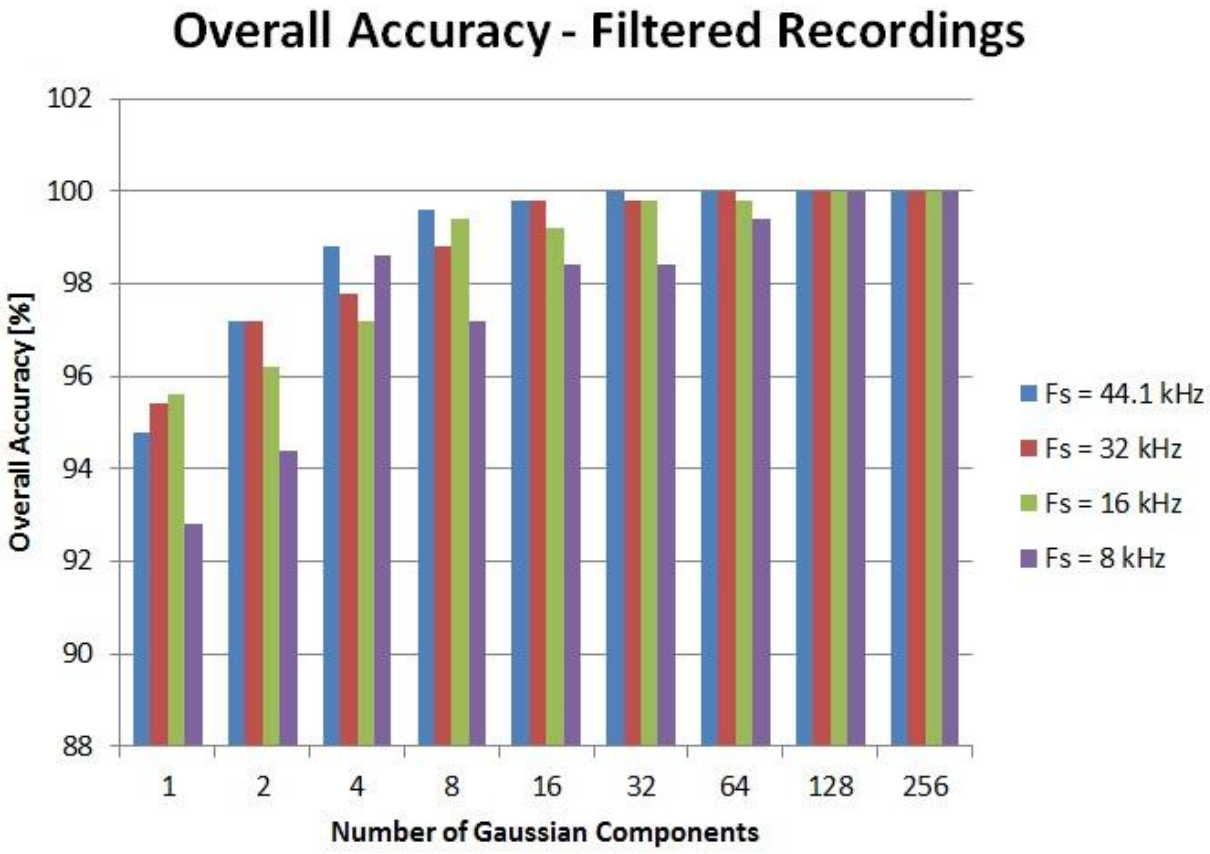

Figure 5.3.2 Overall Accuracy after Filtering

For each sampling frequency, the overall accuracy was presented for filtered and noisy recordings on the same graph:

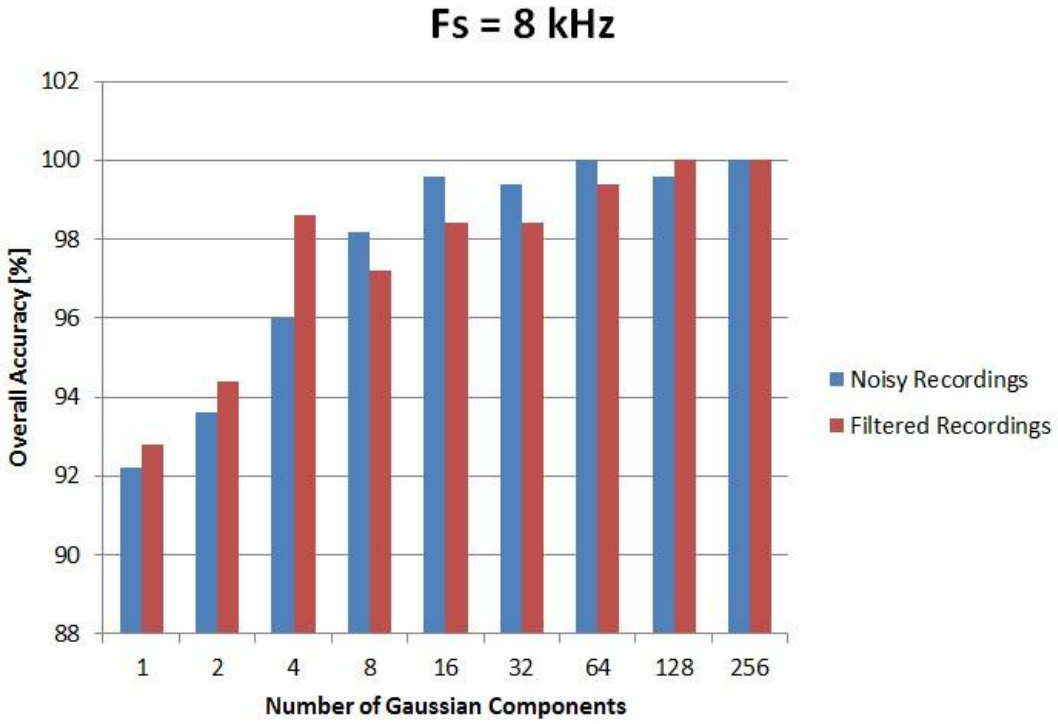

Figure 5.3.3 Overall Accuracy for 8 kHz Sampling Frequency

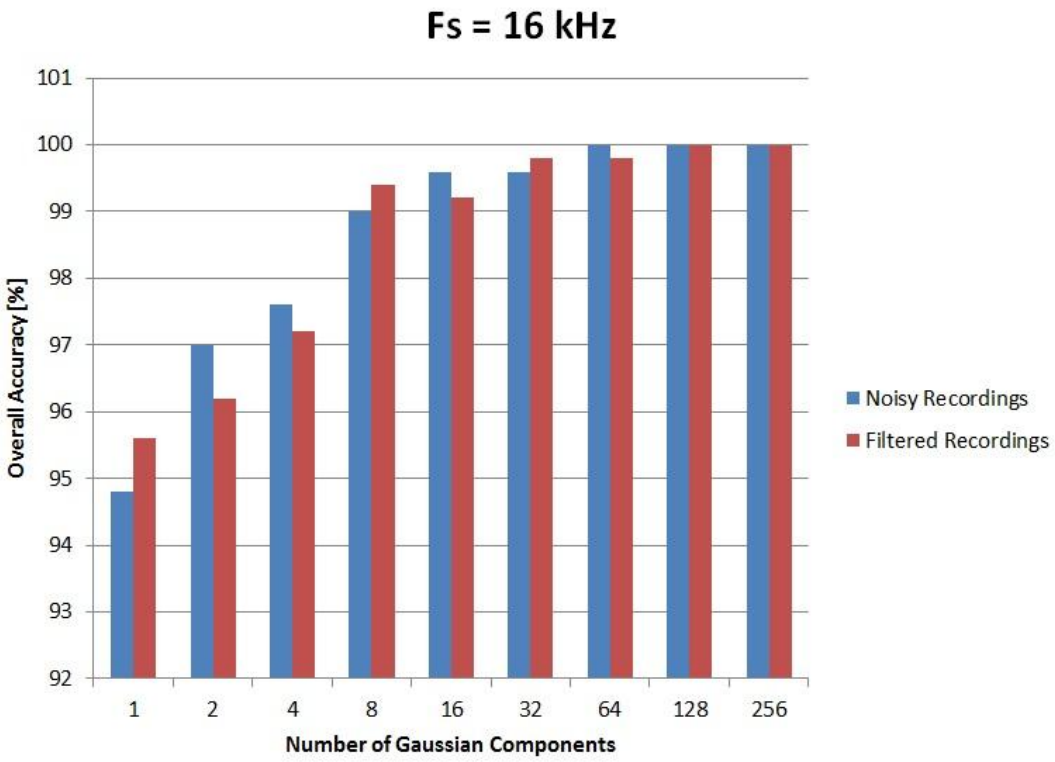

Figure 5.3.4 Overall Accuracy for 16 kHz Sampling Frequency

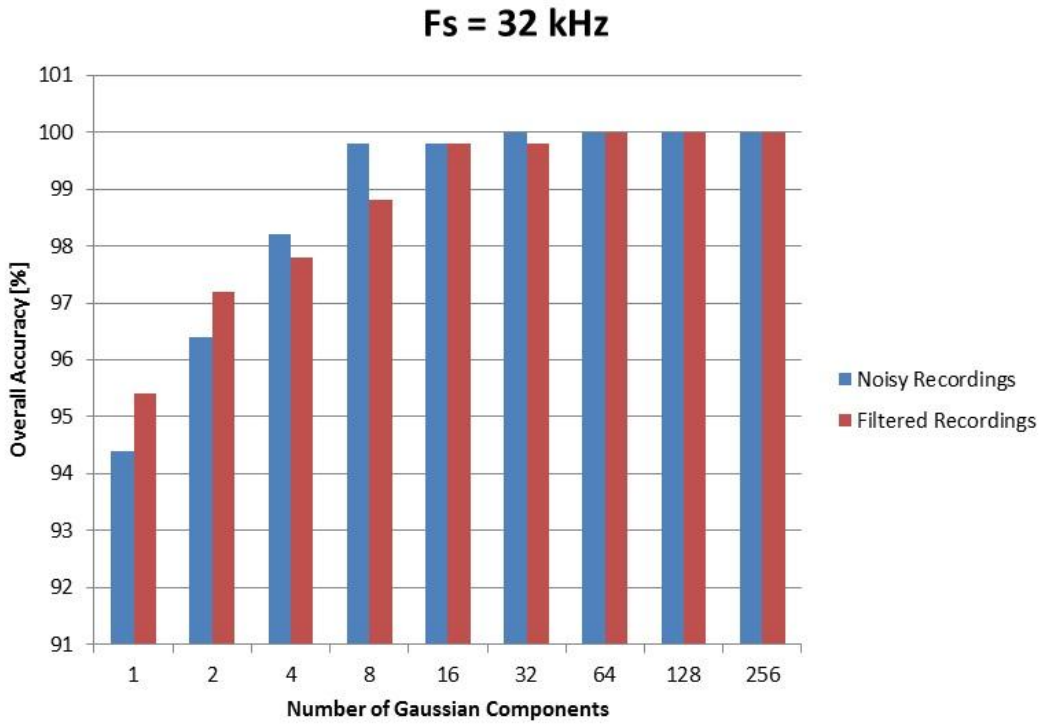

Figure 5.3.5 Overall Accuracy for 32 kHz Sampling Frequency

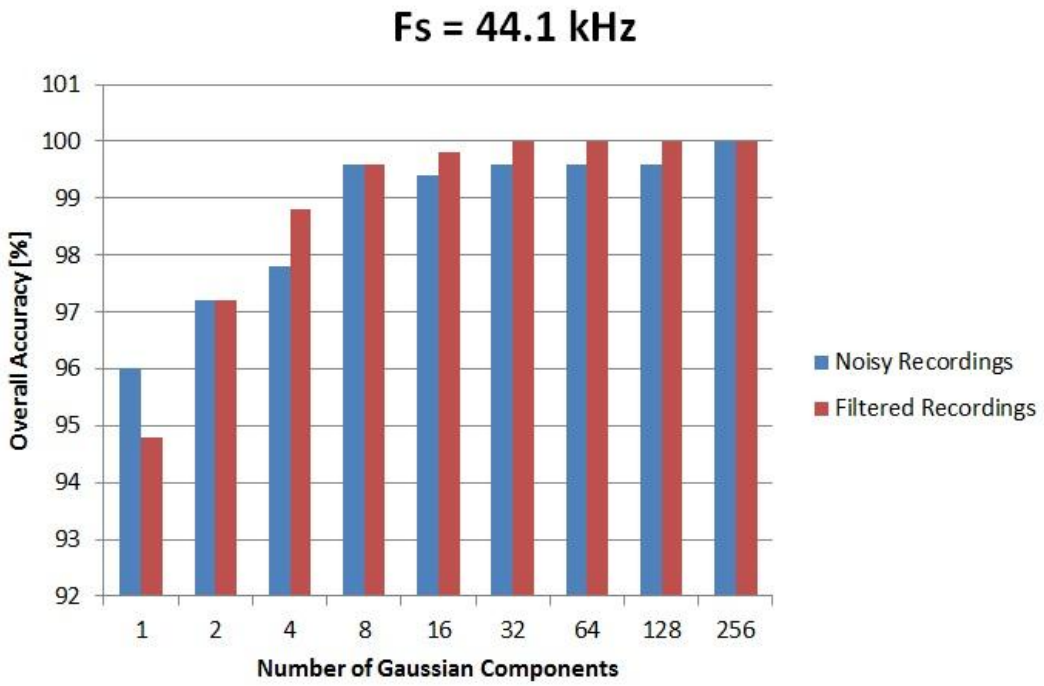

Figure 5.3.6 Overall Accuracy for 44.1 kHz Sampling Frequency

As it can be observed, no improvement is introduced after filtering, so, in order to decrease the latency of the response, this step was eliminated.

CHAPTER 6
CONCLUSION

The main objective of this thesis was to develop a system that recognizes the speaker from a small speaker's set. The project could be divided into three major sections: data acquisition, training and testing.

As stated in the introduction, the purpose of such an approach is to help increase the autonomy of the NAO robot, such that it can interact easily with children. The thesis is desired to be part of a much ampler project that has as goal the introduction in therapy of the robot. This idea is sustained by intensive studies that seem to indicate an increased efficiency in children's therapy when using such a robot. The voice recognition project was developed as it is desired to have a mean of identifying the patients, in order to keep track of their performance. Since the number of children that are part of the study is limited, the speaker's set is initially of only 5 speakers; more will be added if and when necessary. More than that, most of the responses to tasks employed in therapy sessions are based on a given script, so the method used is text dependent.

Some of the most important conclusions that could be drawn from this study are:

- The method is noise robust. Indeed, as it was previously proven, the results are similar for noisy and filtered recordings.
- A compromise between the complexity of the algorithm and the results should be made. The best results are obtained for the highest sampling frequency and the largest number of Gaussian components. But these values (44.1 kHz sampling frequency and 256 Gaussian components) increase not only the memory occupied, but also the response time. Still, very good results are obtained for some lower values for these two parameters.
- The worst case scenario is obtained for a sampling frequency of 8 kHz and only 1 Gaussian component, as it was expected. As observed from the graphical representation previously given, the overall accuracy is still greater than 92%. An explanation for this good result is the low number of speakers, together with the low number of voice commands.

- Another important conclusion that can be drawn is that, for 16 Gaussian components or more, the results in terms of the overall accuracy are similar, regardless of the sampling frequency. Based on this observation, the number of Gaussian components could be decreased, which in turn will lead to a decrease in the response time.

- As it was proven by the values from the confusion matrices presented in the previous chapter, the most mix-ups appear between speaker 4 and speaker 5. Both speakers were female and the most probable explanation for this phenomenon is represented by the similitudes in their voice characteristics.

The personal contributions to this project are:

- The recording of voice commands
- The database creation
- The model training
- The study of the effect of varying sampling frequency and number of Gaussian components
- The study of the effect of filtering on the overall accuracy
- The implementation of the program on the robot

With the technology rapidly evolving, such a project needs to keep up. Some future improvements could be made, especially in the latency decrease direction. Also, if needed, the database could be increased by increasing the number of speakers.

Another aspect that would be an important step towards the autonomy of the robot is the voice activity detection that, if implemented on NAO, would allow it to start acquiring data without any external interference.

REFERENCES

[1] *Humanoid Robot,* https://www.sciencedaily.com/terms/humanoid_robot.htm (accessed on March-12-2018)

[2] Pfeiffer, R., Scheier, C., "Understanding Intelligence", Bradford Books, 2011

[3] *Humanoid robots, learning tools in the world of education,* https://www.ald.softbankrobotics.com/en/solutions/education-research (accessed on March-12-2018)

[4] *NAO,* https://jgrcommunications.com/wp-content/uploads/2016/03/Nao.jpg (accessed on March-12-2018)

[5] *SoftBank Robotics Documentation*, http://doc.aldebaran.com/2-1/home_nao (accessed on March-12-2018)

[6] *LED's position on NAO robot*, http://www.cs.cmu.edu/~cga/nao/doc/reference-documentation/naoqi/sensors/alleds.html (accessed on March-15-2018)

[7] *Wiener filter,* https://en.wikipedia.org/wiki/Wiener_filter (accessed on May-12-2108)

[8] Shimkin, N., "*Estimation and Identification in Dynamical System*", Lecture Notes, fall 2009, Israel Institute of Technology, Department of Electrical Engineering

[9] Dobre, R., "*Multimedia Coding-Techniques and Applications*", Lecture Notes, 2018, Faculty of Electronics, Telecommunications and Information Technology

[10] *Signal Processing*, https://dsp.stackexchange.com (accessed on May-12-2018)

[11] Jurafsky, D., Martin, J.H., "*Speech and Language Processing*", Pearson Education Limited, 2014, p.176-181, 188-194, 298-308, 312-318

[12] *Hidden Markov Models*, https://www.lrde.epita.fr/~reda/cours/speech/PH%20Spoken%20Language%20Processing%20%20A%20Guide%20to%20Theory,%20Algorithm%20and%20System%20Development%20(2001)/chapter8.pdf (accessed on May-12-2018)

[13] Martin, J., "*Natural Language Processing*", Lecture 8, 09-24-2013

[14] *What is a phone,* https://glossary.sil.org/term/phone (accessed on June-02-2018)

[15] *Recognition System of Japanese Characters*, http://www.takanishi.mech.waseda.ac.jp/top/solis/Recognition_Systems/recognition.html (accessed on June-02-2018)

[16] Jurafsky, D., "*Spoken Language Processing*", Stanford University, spring 2014

[17] Jurafsky, D., "*Automatic Speech Recognition*", Stanford University, spring 2014

[18] *Filtering a sound recording*, https://rsmith.home.xs4all.nl/miscellaneous/filtering-a-sound-recording.html (accessed on June-04-2018)

[19] Fayek, H., "*Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepsral Coefficients (MFCCs) and What's in Between*", April-21-2016

[20] *Distributions of One Variable*, http://work.thaslwanter.at/Stats/html/statsDistributions.html (accessed on June-05-2018)

[21] Lee, Jin Long, "*Anomaly Detection, part 2: Multivariate Gaussian Distribution*", Study Notes, June-30-2015

[22] Bhattacharjee, U., Sarmah, K., "*GMM-UBM Based Speaker Verification in Multilingual Environments*", IJCSI International Journal of Computer Science Issues, volume 9, issue 6, no 2, November 2012

[23] Karpov, E., "*Efficient Speaker Recognition for Mobile Devices*", Publications of the University of Eastern Finland Dissertations in Forestry and Natural Sciences, November 2011

[24] Reynolds, D.A., Rose, R.C., "*Robust Text-Independent Speaker Identification Using Gaussian Mixture Speakers Models*", IEEE Transactions on Speech and Audio Processing, volume 3, no 1, January 1995

[25] *Gaussian Mixture Models*, https://prateekvjoshi.com/2013/06/29/gaussian-mixture-models/ (accessed on June-06-2018)

[26] Reynolds, D.A., Quatieri, T.F., Dunn, R.B., "*Speaker Verification Using Adapted Gaussian Mixture Models*", Digital Signal Processing, volume 10, no 1-3, January/April/July 2000

[27] Vuuren, S., "*Speaker Verification in a Time-Feature Space*", Ph.D. thesis, Oregon Graduate Institute, March 1999