ENEKO AÑORGA

DEVELOPMENT OF THE FEATURE EXTRACTOR FOR SPEECH RECOGNITION

DIPLOMA WORK

MARIBOR, OCTOBER 2009

FAKULTETA ZA ELEKTROTEHNIKO,

RAČUNALNIŠTVO IN INFORMATIKO

2000 Maribor, Smetanova ul. 17

Diploma Work for Electronic Engineering Student Program

DEVELOPMENT OF THE FEATURE EXTRACTOR FOR SPEECH RECOGNITION

Student: Eneko Añorga

Student program: Electronic Engineering

Mentor: Prof. Dr. Riko ŠAFARIČ

Asist. Prof. Dr. Suzana URAN

Maribor, October 2009

ACKNOWLEDGMENTS

Thanks to Prof. Dr. Riko ŠAFARIČ for his assistance and helpful advices in carrying out the diploma work.

Special thanks to my family and friends who are in all moments beside me.

DEVELOPMENT OF THE FEATURE EXTRACTOR FOR SPEECH RECOGNITION

Key words: voice operated wheelchair, speech recognition, voice activity detection, neural networks, ultrasound sensor net

UDK: 004.934:681.5(043.2)

Abstract

With this diploma work we have attempted to give continuity to the previous work done by other researchers called, Voice Operating Intelligent Wheelchair – VOIC [1]. A development of a wheelchair controlled by voice is presented in this work and is designed for physically disabled people, who cannot control their movements. This work describes basic components of speech recognition and wheelchair control system.

Going to the grain, a speech recognizer system is comprised of two distinct blocks, a Feature Extractor and a Recognizer. The present work is targeted at the realization of an adequate Feature Extractor block which uses a standard LPC Cepstrum coder, which translates the incoming speech into a trajectory in the LPC Cepstrum feature space, followed by a Self Organizing Map, which classifies the outcome of the coder in order to produce optimal trajectory representations of words in reduced dimension feature spaces. Experimental results indicate that trajectories on such reduced dimension spaces can provide reliable representations of spoken words. The Recognizer block is left for future researchers.

The main contributions of this work have been the research and approach of a new technology for development issues and the realization of applications like a voice recorder and player and a complete Feature Extractor system.

RAZVOJ PREVODNIKA SIGNALA ZA PREPOZNAVO GOVORA

Ključne besede: glasovno voden invalidski voziček, prepoznava govora, zaznava glasovne aktivnosti nevronske mreže, ultrazvočna senzorska mreža

UDK: 004.934:681.5(043.2)

Povzetek

S tem diplomskim delom sem poskusil nadaljevati delo raziskave z naslovom Voice Operating Intelligent Wheelchair – VOIC [1]. V tem diplomskem delu je tudi predstavljen razvoj glasovno vodenega invalidskega vozička, narejenega za telesno prizadete ljudi, ki ne morejo nadzorovati svojih gibov. To delo opisuje osnovne komponente govornega nadzora in sistema vodenja invalidskega vozička.

Sistem govornega nadzora uravnavata dva različna dela; prevodnik signala in prepoznavalec. V tem diplomskem delu se osredotočam na prevodnost ustreznega prevodnika signala na osnovi standardnega LPC Cepstrum koderja, ki posreduje prihajajoči govor v pot LPC Cepstrum prostora, temu postopku pa sledi t.i. "samoorganizacijska karta" (Self Organizing Map), ki razvrsti rezultat koderja za optimalni prikaz besed na zmanjšanih dimenzijah prostora. Poskusni rezultati kažejo, da lahko te poti na zmanjšanih dimenzijah prostora zagotovijo zanesljiv prikaz izgovorjenih besed. Prepoznavalec je lahko predmet raziskovanja študentov tudi v prihodnosti.

Glavni namen tega diplomskega dela sta bili raziskava in poskus uporabe nove tehnologije v razvojne namene, kakor tudi uporaba aplikacij kot so snemalec in predvajalnik zvoka ter celoten sistem prevodnika signala.

Table of Contents

GLOSARY OF SIMBOLS

Name	Description
$1/\hat{A}(z)$	LP synthesis filter
a_i	LP coefficients ($a_0 = 1.0$)
$w_{lp}(n)$	LP analysis window
$H_{h1}(z)$	Input high-pass filter
$s(n)$	Preprocessed/filtered speech signal
$s'(n)$	Windowed speech signal
$r(k)$	Auto-correlation coefficients
$w_{lag}(n)$	Correlation lag window
$r'(k)$	Modified auto-correlation coefficients
k_i	Reflection coefficients
f_s	Sampling frequency
f_0	Bandwidth expansion

Table 1 – Glossary of symbols

GLOSSARY OF ACRONYMS

Acronym	Description
VOIC	Voice Operated Intelligent Wheelchair
DSP	Digital Signal Processor
LPC	Linear Prediction Coding
LP	Linear Prediction
SOM	Self Organizing Maps
CE	Compact Edition
SODIMM	Small Outline Dual In-line Memory Module
CAN	Control Area Network
GPIO	General Purpose Input/Output
BSP	Board Support Package
ITU	International Telecommunication Union
ITU-T	Telecommunication Standardization Sector
FE	Feature Extractor
VAD	Voice Activity Detection
DTX	Discontinuous Transmission
CNG	Comfort Noise Generator
RNN	Recurrent Neural Network
VQ	Vector Quantization
HMM	Hidden Markov Model

Table 2 – Glossary of acronyms

1 INTRODUCTION

1.1 MOTIVATION

A prototype of a wheelchair controlled by voice was developed by previous researchers. The wheelchair was designed for physically disabled people, who have problems with mobility and are not able to use some mechanical devices like, for example, a joystick. For that reason the researchers worked in other way to control the wheelchair: the voice.

The beginning of the project was in September 2003. Just to get an idea of how extensive the work was, in the project were involved 11 students of Mechatronics and Computer Science, of which 5 were working in the first part and 6 in the second one. The project was successfully completed in one year.

The first part consisted in a production of a specific control module for the management of the wheelchair and the production of an adequate system of an ultrasound sensor net to capture data from the surroundings, such as location and distance barriers.

The second part was oriented to identify the voice commands, building a speech recognition system to control the wheelchair.

The main goal of the project was to recognize the user's speech in different environments and consequently control the wheelchair. The speech recognition task is natural and easy for humans, but it is still a difficult job for computers and no perfect solution has been found until now.

In general terms, the basic principles of the wheelchair operating using a recognized voice commands and the ultrasound sensor net system are shown in the Figure 1.1 and described in the following lines:

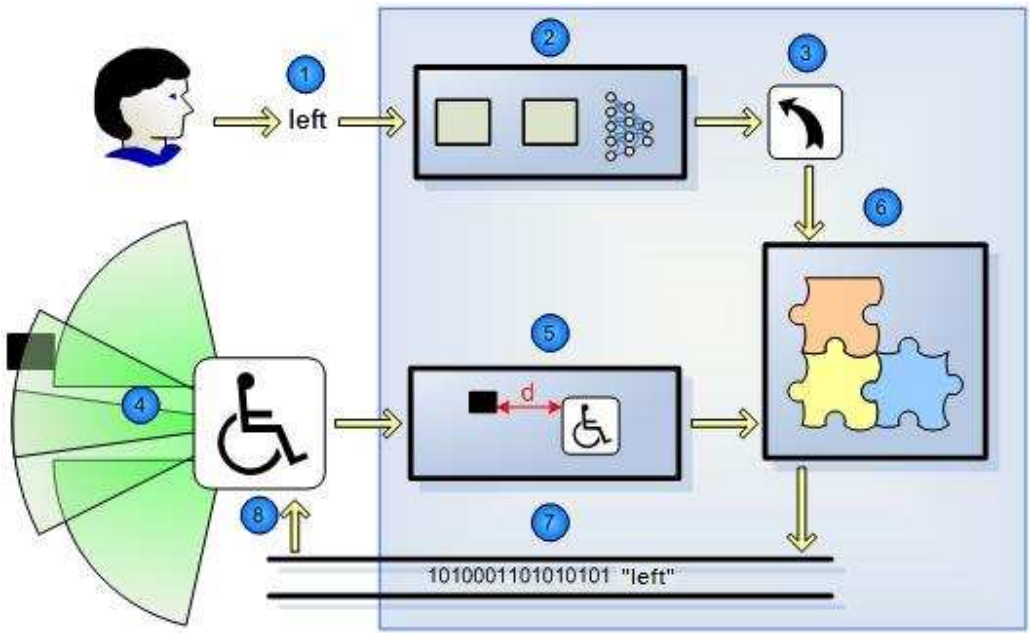

Figure 1.1 – Principle operation of the wheelchair

1. A wheelchair user utters a command.
2. The speech recognition system processes the speech signal, analyzes, reduces the coefficients dimensions and recognizes the signal trajectory. The result is a suitable coded recognized command.
3. The command is transferred to the part of the system designed for its evaluation and execution.
4. Wheelchair surrounding dynamical obstacles are being observed all the time during its operation by the ultrasound sensor net measuring the distance to the obstacles.
5. A special designed component measures distance to the obstacle.
6. A control system collects all necessary data and decides whether it is safe to execute recognized the voice command.

7. In the case that the recognized command implies no threat to the wheelchair user and people surrounding it, the command is transmitted to the wheelchair for execution.

8. The wheelchair receives the command and executes it.

As seen in the earlier description, all the tasks were centralized in a control system and the whole system was implemented specifically in a DSP Card TMS320C6711DSK from *Texas Instruments* manufacturer. For programming the DSP processor the Code Composer Studio software was used and also the Visual Studio .NET 2003 for C language programming environment.

Till this point, all was ok but the problem came when the DSP Card was broken. At this point the researchers thought of changing it for a new DSP Card but they realized that it was already obsolete technology. Hence, a decision was taken: replace the DSP Card for a new device, which was able to perform the same task and was equipped with all necessary interfaces, like audio input to capture the voice signal and then make the speech recognition task or CAN bus for the ultrasound sensor system. In addition, this device would have to be able to control different systems, easy to use, small size to plug into the wheelchair, economical...

Here is where our work began, so let's see which have been our objectives and contributions to the project.

1.2 OBJECTIVES AND CONTRIBUTION OF THIS WORK

As mentioned before, the aim of the work was to replace the broken DSP Card by a new device which was able to control the whole system and which had some other important features like an audio input interface (microphone) to process the user's voice signal, a CAN bus interface to control the ultrasound sensors net; it had to be scalable, a new technology, easy to use, small size, economical...

After an in-depth research we found Colibri XScale® PXA320 computer module which contains all these features together, which will they will be explained in next chapters (*2.1 Brief description of Colibri module*).

Mainly the contribution of this work has been to adapt the work which was already developed to the new technology recently obtained.

At this point, we have to say that this was not an easy task, specifically trying to adapt the developed code. As all the C code developed was targeted to the DSP Card, we had to start almost from the beginning to adapt it to the new device and new technology. In addition, knowing that the previous work was made by 11 students and the present work has been done by one person, we focus our objective on solving a part of the speech recognition system, and most particularly, on realizing the Feature Extraction block.

In a speech recognition problem the FE block has to process the incoming information, the speech signal, so that its output eases the work of the recognition stage. The approach used in this work to design the FE block divides it into two consecutive sub-blocks: the first is based on speech coding techniques, and the second uses a SOM for further optimization (data dimensionality reduction). We will see the development of this block in more detail in the section *2.2 Design of the Feature Extraction*.

Summarizing, the main contributions of this work are the following:

- The first one was to make a market research to find a technology that would meet our needs. We found it and also we prepared it for an adequate development environment.

- Then, we built a complete audio Recorder and Player. This task was not strictly necessary but the aim of this has been to learn, practice and improve the C ++ programming skills. With this audio Recorder and Player we are able to record the voice during some time, with different sample rates, different resolutions and finally we can save it in a .wav file for next processing steps like the speech coding. In addition we will be able to play and listen to the recorded signals.

- Finally, we made the FE block for the speech recognition system, based on the ITU-T's G.729 Recommendation. This Recommendation contains the description of an algorithm for the coding of speech signals using Linear Prediction Coding. This Recommendation also includes an electronic attachment containing reference C code which we used as a reference to build our own software getting satisfying results.

1.3 ORGANIZATION OF THIS WORK

The hardware and software used for the implementation of the work is detailed in first place. Then, the implementation of the FE block is presented, which is the major research of this work and is presented following the conceptual division stated in Figure 2.5. The work concludes by commenting on the results and proposing future steps for the continuing of the project.

1.4 RESOURCES

The resources used for this work, apart from obviously the infinite amount of information found in the internet to answer several questions, papers and articles..., were basically, the Colibri computer module and computational resources like a normal PC to generate the software for it. We did the programming tasks in C++ language using Microsoft embedded Visual C++ for developing environment. Also we used Matlab suite to contrast results.

2 DEVELOPMENT

2.1 BRIEF DESCRIPTION OF COLIBRI MODULE

2.1.1 Hardware

The Swiss company Toradex AG is headquartered in Horw nearby the city of Lucerne, and maintains local customer support offices in several countries throughout Europe and North America. Toradex® is specialized in highly miniaturized embedded computers based on Marvell® XScale® (Bulverde PXA270, Monahans PXA3xx) and X86 (Intel® Atom®) Processors [9].

For our purposes we have used one of its products, exactly, the Colibri XScale® PXA320 which is a SODIMM sized computer module (See Figure 2.1). Its processor runs at up to 806 MHz and consumes about 800 mW. The module's targets are the low power systems that still require high CPU performance.

It also offers all the interfaces needed in a modern embedded device: beside the internal Flash memory, there are plenty of interfaces available for data storage: Compact Flash/ PCMCIA and SD Card. The module provides glueless connectivity to passive and active LCDs with resolutions of up to 1024x768, as well as 4-wire resistive touch screens. An integrated 16 bit stereo codec allows Colibri PXA320 to play and record sound. Colibri PXA320 can directly connect to a CMOS/CCD camera sensor. In addition Colibri PXA320 offers a 100 Mbit Ethernet connection as well as an USB host and USB device functionality.

Figure 2.1 – Colibri PXA320

Module Specifications:

CPU

PXA320 806MHz

Memory

128MB DDR RAM (32Bit)

1GB NAND Flash (8Bit)

Interfaces

16Bit External BUS

Compact Flash/PCMCIA

LCD (SVGA)

Touch Screen

Audio I/O (16Bit Stereo)

CMOS image sensor

I2C

SPI

2x SD Card

USB Host/Device

100MBit Ethernet

2x UART

IrDA

PWM

127 GPIOs

Software

Pre-installed

Windows CE 5.0/6.0

Size

67.6 x 36.7 x 5.2 mm

Temperature Range

0 to +70°C

-45 to +85°C (IT version)

In order to have a flexible development environment to explore the functionality and performance of the Colibri modules the Colibri Evaluation Board is used (See Figure 2.2).

Besides the user interfaces it provides numerous communication channels as well as a configurable jumper area to hook up the Colibri GPIOs to the desired function. To facilitate interfacing to the custom hardware the Colibri Evaluation Board provides the buffered CPU bus on a separate connector.

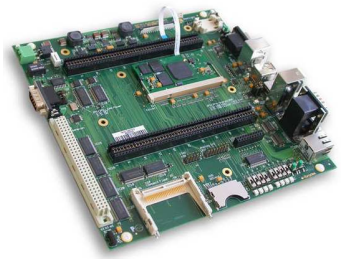

Figure 2.2 – Colibri Evaluation Board

Module Specifications:

CPU Modules

Colibri PXA270

Colibri PXA300

Colibri PXA310

Colibri PXA320

Interfaces

10/100MBit Ethernet

USB Host/Device

USB Host

2x PS/2

Analogue VGA

Generic LCD Connector

TFT: Philips LB064V02-A1

Line-In, Line-Out, Mic-In

IrDA

2x RS232

CAN (Philips SJA1000)

SD Card

Compact Flash

Power Supply:

Required Input:

7-24VDC, 3-50W

On-board Converter:

3.3V, 5V max 5A

Size:

200 x 200 mm

The received invoice from Toradex, for the Colibri XScale® PXA320, plus the Colibri Evaluation Carrier Board and plus the support hours are shown in the next Figure 2.3:

Pos	Description	Units	Unit Price	Price
1	Colibri XScale PXA320IT 806MHz (Embedded Computer Module, SODIMM form factor) Scheduled Delivery 4 weeks after received payment	1	€ 149.00	€ 149.00
2	Colibri Evaluation Carrier Board for use with the entire Colibri Product Family Scheduled Delivery 4 weeks after received payment	1	€ 299.00	€ 299.00
3	Support hours valid during 6 months	1	€ 125.00	€ 125.00
	TOTAL GOODS AND SERVICES			€ 573.00

Figure 2.3 – Invoice from Toradex

2.1.2 Software

The module is shipped with a preinstalled WinCE 5.0 image with WinCE Core license. Other OS like Embedded Linux are available from the third-party.

Toradex provides a WinCE 5.0 image and a WinCE 6.0. All WinCE images contain the Toradex Board Support Package (BSP) which is one of the most advanced BSPs available on the market. Besides the standard Windows CE functionality, it includes a large number of additional drivers as well as optimized versions of standard drivers for the most common interfaces and is easily customizable by registry settings to adapt to specific hardware.

The Microsoft® eMbedded Visual C++ 4.0 tool is used as desktop development environment for creating the applications and system components for Windows® CE .NET powered devices.

In conclusion, all the software presented in this work was done using the Microsoft® eMbedded Visual C++ 4.0 development tool and the Toradex BSP tool (Figure 2.4).

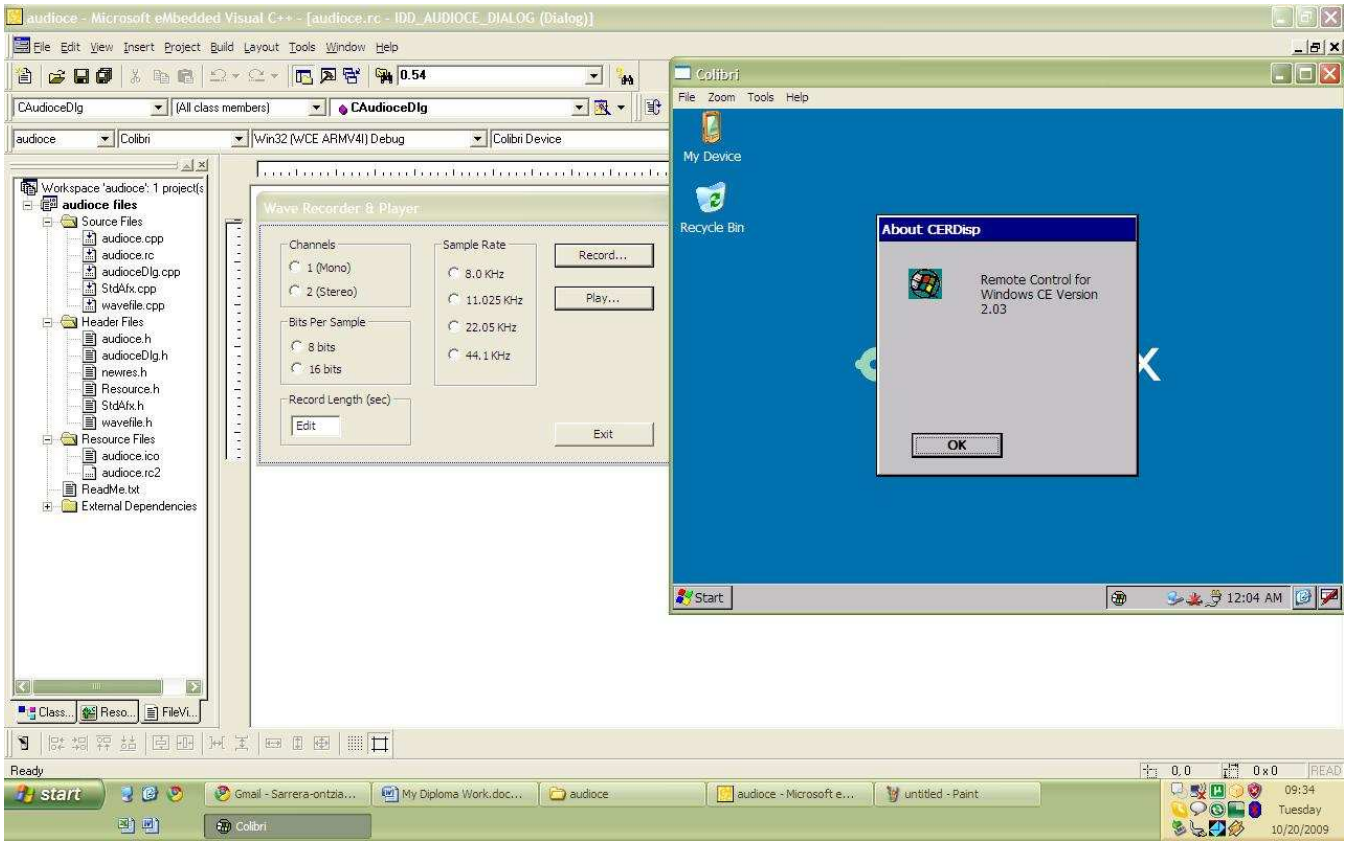

Figure 2.4 – Microsoft® eMbedded Visual C++ 4.0 and Windows® CE

2.2 DESIGN OF THE FEATURE EXTRACTOR

As stated before, in a speech recognition problem the FE block has to process the incoming information, the speech signal, so that its output eases the work of the classification stage. The approach used in this work designs the FE block and divides it into two consecutive sub-blocks: the first is based on speech coding techniques, and the second uses a SOM for further optimization (data dimensionality reduction). The different blocks and sub-blocks are shown in the next Figure 2.5:

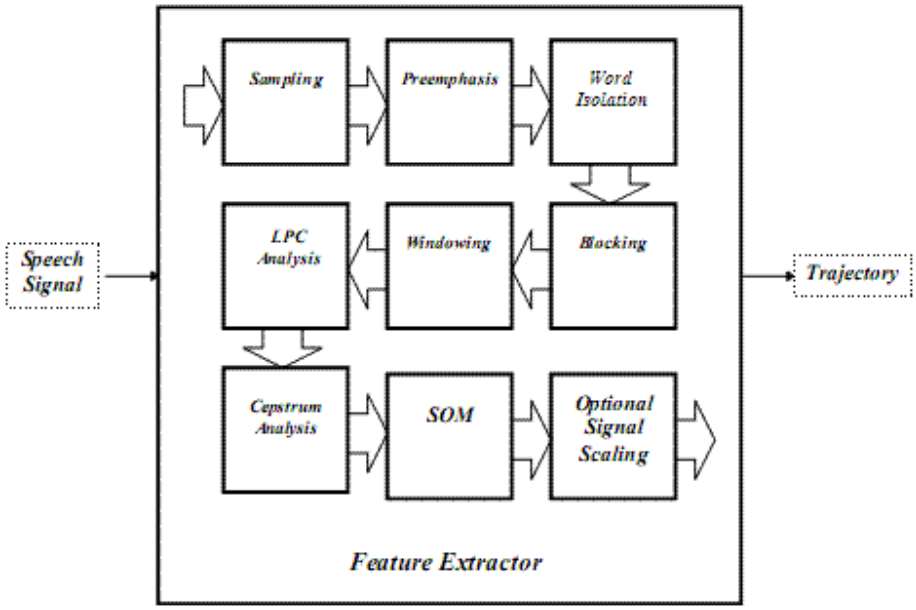

Figure 2.5 – FE schematic

2.2.1 Speech coding

2.2.1.1 Speech sampling

The speech was recorded and sampled using a relatively inexpensive dynamic microphone and a Colibri's audio input interface. The incoming signal was sampled at 8.000 Hz with 16 bits of resolution.

It might be argued that a higher sampling frequency, or more sampling precision, is needed in order to higher recognition accuracy. However, if a normal digital phone, which samples speech at 8.000 Hz with a 16 bit resolution, is able to preserve most of the information carried by the signal [6], it does not seem necessary to increase the sampling rate beyond 8.000 Hz or the sampling precision to something higher than 16 bits. Another reason behind these settings is that

commercial speech recognizers typically use comparable parameter values and achieve impressive results.

2.2.1.2 Pre-emphasis filter

After sampling the input signal is convenient to filter it with a second order high-pass filter with cut off frequency at 140 Hz. The filter serves as a precaution against undesired low-frequency components.

The resulting filter is given by:

$$H_{h1}(z) = \frac{0.46363718 - 0.92724705z^{-1} + 0.46363718z^{-2}}{1 - 1.9059465z^{-1} + 0.9114024z^{-2}} \quad (1)$$

2.2.1.3 Word Isolation

Despite the fact that the sampled signal had pauses between the utterances, it was still needed to determine the starting and ending points of the word utterances in order to know exactly the signal that characterized each word. To accomplish this, we decided to use VAD (Voice Activity Detection) technique used in speech processing, instead of using the rolling average and the threshold, determined by the start and end of each word, used in the previous works, with the aim of achieving more accuracy and efficiency.

For that, we based our work in the Annex B from the ITU's Recommendation G.729 [5], where a source code in C language about the VAD is efficiently developed.

VAD is a method which differentiates speech from silence or noise signal to aid in speech processing and the Annex B provides a high level description of the Voice Activity Detection (VAD), Discontinuous Transmission (DTX) and Comfort Noise Generator (CNG) algorithms. These algorithms are used to reduce the transmission rate during silence periods of speech. They

are designed and optimized to work in conjunction with [ITU-T V.70]. [ITU-T V.70] mandates the use of speech coding methods. The algorithms are adapted to operate with both the full version of G.729 and Annex B.

Let's see a general description of the VAD algorithm:

The VAD algorithm makes a voice activity decision every 10 ms in accordance with the frame size of the pre-processed (filtered) signal. A set of difference parameters is extracted and used for an initial decision. The parameters are the full-band energy, the low-band energy, the zero-crossing rate and a spectral measure. The long-term averages of the parameters during non-active voice segments follow the changing nature of the background noise. A set of differential parameters is obtained at each frame. These are a difference measure between each parameter and its respective long-term average. The initial voice activity decision is obtained using a piecewise linear decision boundary between each pair of differential parameters. A final voice activity decision is obtained by smoothing the initial decision.

The output of the VAD module is either 1 or 0, indicating the presence or absence of voice activity respectively. If the VAD output is 1, the G.729 speech codec is invoked to code/decode the active voice frames. However, if the VAD output is 0, the DTX/CNG algorithms described herein are used to code/decode the non-active voice frames.

2.2.1.4 Speech coding

After the signal was sampled, the spectrum was flattened, and the utterances were isolated we tried to codify it using the Linear Prediction Coding (LPC) method [3].

In a variety of applications, it is desirable to compress a speech signal for efficient transmission or storage. For example, to accommodate many speech signals in a given bandwidth of a cellular phone system, each digitized speech signal is compressed before transmission. For medium or low bit-rate speech coders, LPC method is most widely used. Redundancy in a speech signal is removed by passing the signal through a speech analysis filter.

The output of the filter, termed the residual error signal, has less redundancy than the original speech signal and can be quantized by a smaller number of bits than the original speech.

The short-term analysis and synthesis filters are based on 10th order linear prediction (LP) filters.

The LP synthesis filter is defined as:

$$\frac{1}{\hat{A}(z)} = \frac{1}{1+\sum_{i=1}^{10} \hat{a}_i z^{-i}} \tag{2}$$

where $\hat{a}_i$, i = 1,...,10, are the quantized Linear Prediction (LP) coefficients. Short-term prediction or linear prediction analysis is performed once per speech frame using the autocorrelation method with a 30 ms (240 samples) asymmetric window. Every 10 ms (80 samples), the autocorrelation coefficients of windowed speech are computed and converted to the LP coefficients using the Levinson-Durbin algorithm. Then the LP coefficients are transformed to the LSP domain for quantization and interpolation purposes. The interpolated quantized and unquantized filters are converted back to the LP filter coefficients (to construct the synthesis and weighting filters for each subframe).

The LP analysis window consists of two parts: the first part is half a Hamming window and the second part is a quarter of a cosine function cycle. The window is given by:

$$w_{lp}(n) = \begin{cases} 0.54 - 0.46\cos\left(\dfrac{2\pi n}{399}\right) & n = 0,...,199 \\ \cos\left(\dfrac{2\pi(n-200)}{159}\right) & n = 200, \ ... \ 239 \end{cases} \tag{3}$$

There is a 5 ms look-ahead in the LP analysis which means that 40 samples are needed from the future speech frame. This translates into an extra algorithmic delay of 5 ms at the encoder stage. The LP analysis window applies to 120 samples from past speech frames, 80 samples from

the present speech frame, and 40 samples from the future frame. The windowing procedure is illustrated in Figure 2.4.

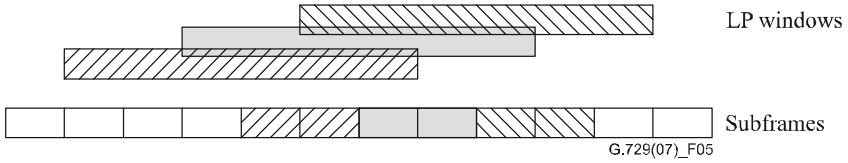

Figure 2.6 – Windowing procedure in LP analysis

The different shading patterns identify corresponding excitation and LP analysis windows.

The windowed speech:

$$s'(n) = w_{lp}(n)\, s(n) \quad n = 0,...,239 \tag{4}$$

is used to compute the autocorrelation coefficients:

$$r(k) = \sum_{n=k}^{239} s'(n)\, s'(n-k) \quad k = 0,...,10 \tag{5}$$

To avoid arithmetic problems for low-level input signals the value of $r(0)$ has a lower boundary of $r(0) = 1.0$. A 60 Hz bandwidth expansion is applied by multiplying the autocorrelation coefficients with:

$$w_{lag}(k) = exp\left[-\frac{1}{2}\left(\frac{2\pi f_0 k}{f_s} \right)^2 \right] \quad k = 1,...,10 \tag{6}$$

where $f_0 = 60$ Hz is the bandwidth expansion and $f_s = 8000$ Hz is the sampling frequency. Furthermore, $r(0)$ is multiplied by a white-noise correction factor 1.0001, which is equivalent to adding a noise floor at –40 dB. The modified autocorrelation coefficients are given by:

$$\begin{aligned} r'(0) &= 1.0001\, r(0) \\ r'(k) &= w_{lag}(k)\, r(k) \quad k = 1,...,10 \end{aligned} \tag{7}$$

The modified autocorrelation coefficients $r'(k)$ are used to obtain the LP filter coefficients, a_i, $i = 1,...,10$, by solving the set of equations:

$$\sum_{i=1}^{10} a_i r'(|i-k|) = -r'(k) \quad k = 1,...,10 \tag{8}$$

The set of equations in (8) is solved using an efficient algorithm known as Levinson-Durbin algorithm. This algorithm uses the following recursion:

$E^{[0]} = r'(0)$
for $i = 1$ to 10
  $a_0^{[i-1]} = 1$
  $k_i = -\left[\sum_{j=0}^{i-1} a_j^{[i-1]} r'(i-j)\right] / E^{[i-1]}$
  $a_i^{[i]} = k_i$
  for $j = 1$ to $i-1$
    $a_j^{[i]} = a_j^{[i-1]} + k_i a_{i-j}^{[i-1]}$
  end
  $E^{[i]} = (1 - k_i^2) E^{[i-1]}$
end

The final solution is given as $a_j = a_j^{[10]}$, $j = 0...10$, with $a_0 = 1.0$.

Finally, the LPC Cepstrum coefficients were obtained using:

$$c_m = \begin{cases} a_m + \sum_{k=1}^{m-1} \frac{k c_k a_{m-k}}{m} & , \quad m \in [1, p] \\ \sum_{k=1}^{m-1} \frac{k c_k a_{m-k}}{m} & , \quad m > p \end{cases}$$

where a value of m = 10 was used in the feature extractor, resulting in 10 LPC Cepstrum values per frame. As a result of the LPC Cepstrum extraction procedure, each utterance was translated into a sequence of points, each belonging to the LPC Cepstrum feature space of dimension 10, and each 10 ms apart. In other words, this procedure translates an air pressure wave into a discretized trajectory in the LPC Cepstrum feature space.

2.2.1.5 Summing up

The speech coding different stages are the following:

1. Sampling: the voice signal is sampled at 8.000 Hertz with 16 bits of resolution.
2. Pre-emphasis filter: the sampled signal is filtered by a second order high-pass filter.
3. Word Isolation: the filtered signal is passed through the VAD block to isolate the word.
4. Blocking: the isolated word is divided into a sequence of data blocks of fixed length, called frames and multiplied by Hamming window of same width.
5. LPC analysis: for each frame, 10 LPC coefficients are calculated.
6. Cepstrum analysis: the 10 LPC coefficients are converted in 10 Cepstral coefficient ones, which are the output from the speech coding sub-block and the input of next one: the Dimensionality reduction.

2.2.2 Dimensionality reduction using SOM

The Self Organizing Map (SOM) is a neural network that acts like a transform which maps an m-dimensional input vector into a discretized n-dimensional space while locally preserving the topology of the input data [4]. The expression “locally preserving the topology” means that for certain volume size in the input space, points that are close together in the input space correspond to neurons that are close in the output space. This is the reason that explains why a SOM is called a feature map: relevant features are extracted from the input space and presented in the output space in an ordered manner. It is always possible to reverse the mapping and restore the original set of data to the original m-dimensional space with a bounded error. The bound on this error is determined by the architecture of the network and the number of neurons. The SOM considers the data set as a collection of independent points and does not deal with the temporal characteristics of the data. It is a very special transform in the sense that it re-expresses the data in a space with a different number of dimensions while preserving some part of the topology.

A typical Kohonen SOM architecture is shown below (Figure 2.7). It consists of an input layer connected to an output layer (two-dimensional Kohonen layer) via a Kohonen Synapses. Each neuron in a Kohonen Layer is associated with a unique set of co-ordinates in two-dimensional space, and hence is referred to as a Position Neuron. The input layer with 'n' input neurons is fed with n-dimensional input data one by one. The output layer organizes itself to represent the inputs.

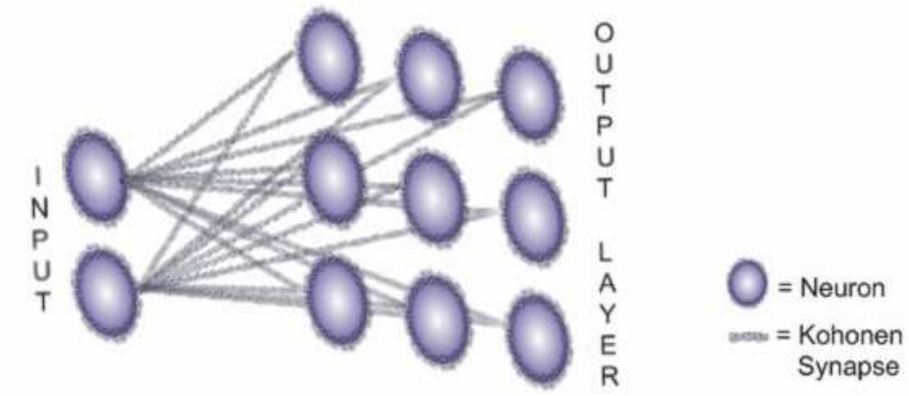

Figure 2.7 – A typical Kohonen SOM network

During the training phase, a SOM builds a representation of training samples. The trained network can be used to map any input vector onto two-dimensional space.

The objective of SOM training is to ensure that different parts of the network respond similarly to similar input vectors. So, the training mainly involves analysing the behaviour of the network for a training sample and adjusting the weights of synapses to ensure that the network exhibits a similar behaviour for a similar input.

The training procedure involves the following steps:

1. The neurons are arranged in an n-dimensional lattice. Each neuron stores a point in an m-dimensional space.

2. A randomly chosen input vector is presented to the SOM. The neurons start to compete until the one that stores the closest point to the input vector prevails. Once the dynamics of the network converge, all the neurons but the prevailing one will be inactive. The output of the SOM is defined as the co-ordinates of the prevailing neuron in the lattice.

3. A neighbourhood function is centred on the prevailing neuron of the lattice. The value of this function is one at the position of the active neuron, and decreases with the distance measured from the position of the winning neuron.

4. The points stored by all the neurons are moved towards the input vector in an amount proportional to the neighbourhood function evaluated in the position of the lattice where the neuron being modified stands.

5. Return to 2, and repeat steps 2, 3, and 4 until the average error between the input vectors and the winning neurons reduces to a small value.

After the SOM is trained, the co-ordinates of the active neuron in the lattice are used as its outputs.

Once the utterance is translated into a trajectory the word recognition problem becomes into a trajectory recognition problem. In this approach the dimensionality of the trajectories is

reduced before feeding them into the Recognizer block. In this manner, the trajectory classification is highly simplified.

Even more, despite the fact that the utterances are produced by a biological system; a system that necessarily produces continuous outputs, different utterances can represent the same word. It is important to note that each of these alternate utterances is valid and none of them can be regarded as a deviation from some ideal way to enunciate that word or as an incorrect output. In other words, there is no one-to-one relationship between the set of possible utterances and the class to which they belong: one utterance necessarily implies only one class, but a class does not necessarily imply only one utterance. This aspect makes any analytical representation of the problem more complex than it could be expected.

The most common approach is to use the obtained trajectory to generate a sequence of labels, normally by means of a Vector Quantization (VQ) scheme [7]. This sequence is then used for recognition. As an example, Carnegie Mellon's SPHINX speech recognition system [8] fed the output of the speech coding scheme into a VQ system which translated the incoming data into a sequence of phonemes. The SPHINX system then used an HMM approach to process the sequences of labels and recognize the words.

Using the fact that the SOM is a VQ scheme that preserves some of the topology in the original space, the basic idea behind the approach employed in this work is to use the output of a SOM trained with the output of the LPC Cepstrum block to obtain reduced state space trajectories that preserve some of the behaviour of the original trajectory. The problem is now reduced to find the correct number of neurons for constituting the SOM and their geometrical arrangement.

The SPHINX speech recognition system quantized the trajectories, which belonged to a space of 12 dimensions, using a VQ scheme with 256 vectors to generate a sequence of labels [8]. The SPHINX system achieved high recognition accuracy in a much more difficult problem than the one being investigated in this work. It consisted of recognizing words from a 1000 word vocabulary, under continuous speech conditions, and in the presence of multiple speakers. Based on these results, since a SOM is a VQ scheme that preserves some part of the topology of the original space, a SOM with as many neurons as vectors the SPHINX system had in its codebook

set should be capable of an efficient quantization of the input space. In other words, it should be capable of at least retaining the amount of information needed to achieve the recognition accuracy reached by the SPHINX system.

Based on the ideas stated above, it was decided that a SOM with 256 neurons was enough to reduce the dimensionality of the trajectories while keeping enough information to achieve high recognition accuracy. The SOM was arranged in a two-dimensional lattice. The co-ordinates of the lattice were used as the co-ordinates of the reduced space trajectory. The fact that the outcome is a two-dimensional trajectory which can be graphically represented for visual display was not particularly important for our decision about the number of dimensions of the output space.

It must be noted that a similar approach was used by Kohonen, but instead of reducing the dimensionality of the trajectory, he used the SOM to generate a sequence of labels, which was then used by a word classifier. Thus the present use of the SOM for order reduction in trajectory representations is a novel one and this application can find appropriate use in other problems besides speech recognition where similar trajectories arise [2].

2.2.2.1 Optional Signal Scaling

The output of the SOM is defined by the co-ordinates of the neuron that was activated by the input. The output values may need to be scaled before they are fed into the Recognizer.

2.2.2.2 Summing up

As it was defined before, the Feature Extractor block works as a transducer that translates the information contained by an air pressure wave into a trajectory in some feature space.

Figure 2.5 sums up the processing done by the FE block. The different steps follow one another according to the following sequence:

1. The incoming pressure wave is reduced to a digital signal through a sampling process.
2. The starting and the ending points of the utterance embedded into the signal are obtained using the VAD process.
3. The spectrum of the extracted utterance is enhanced by means of a pre-emphasis filter which boosts the high frequency components.
4. Several data blocks are extracted from the enhanced signal.
5. The extracted data blocks are windowed to reduce leakage effects.
6. LPC components are extracted from the filtered blocks.
7. LPC Cepstrum components are then extracted from the LPC vectors.
8. The dimensionality of the LPC Cepstrum vectors is reduced using a SOM.
9. The resulting vectors are scaled if the Recognizer requires it.

Nothing can still be said about the overall effectiveness of the FE block, since it depends on the recognition accuracy of the overall system. As an example, if the complete system achieves low recognition percentages, that can be caused by the FE block or the Recognizer, but, if it achieves higher percentages that means that the FE block was at least able to produce data that allowed these recognition accuracies. In other words, in order to know the usefulness of the FE block, the Recognizer outputs must be obtained first.

2.3 SOFTWARE DEVELOPMENT

2.3.1 Audio Recorder and Player

As we have mentioned before, we built a complete audio Recorder and Player. This task was not strictly necessary but the aim of this has been to learn, practice and improve the C ++ programming skills. The graphic interface of the program is shown in the following Figure 2.8:

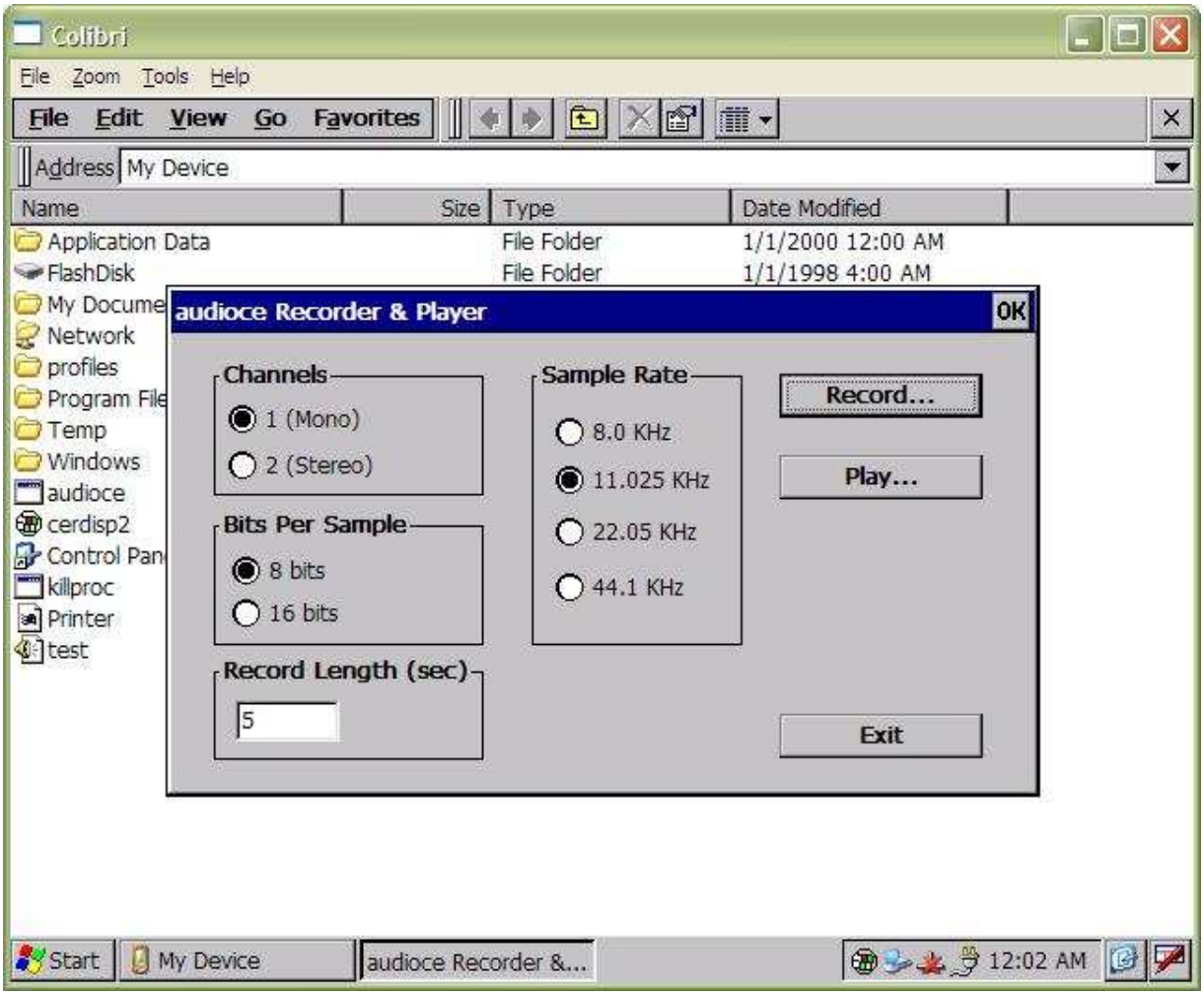

Figure 2.8 – audioce Recorder & Player

As we can see in the Figure 2.8, with this software we are able to record the voice during some time, with different sample rates, different resolutions and finally we can save it in a .wav file for next processing steps like the speech coding. In addition, we will be able to play and listen to the recorded signals. Along with the documentation (Annex 2) is included an electronic attachment containing the source code in C++ used to build the software and which is coming with all the necessary explanations.

As we can see in the next Figure 2.9 we can see the difference between the different ways of sampling. In general, the memory occupied by the sound file is proportional to the number of samples per second and the resolution of each sample. For the first case the speech is sampled at 8.0 KHz and with 8 bits of resolution (1 byte/sample), this means that the memory occupied for the sound file will be 8.000 (samples/sec) x 5 (sec) = 40.000 (samples) = 40.000 (samples) x 1 (byte/sample) = 40 Kbytes.

For the second case the speech is sampled at 44.1 KHz and with 16 bits of resolution (2 bytes/sample), the memory occupied for the sound file will be 44.100 (samples/sec) x 5 (sec) = 220.500 (samples) = 220.500 (samples) x 2 (bytes/sample) = 440.1 Kbytes.

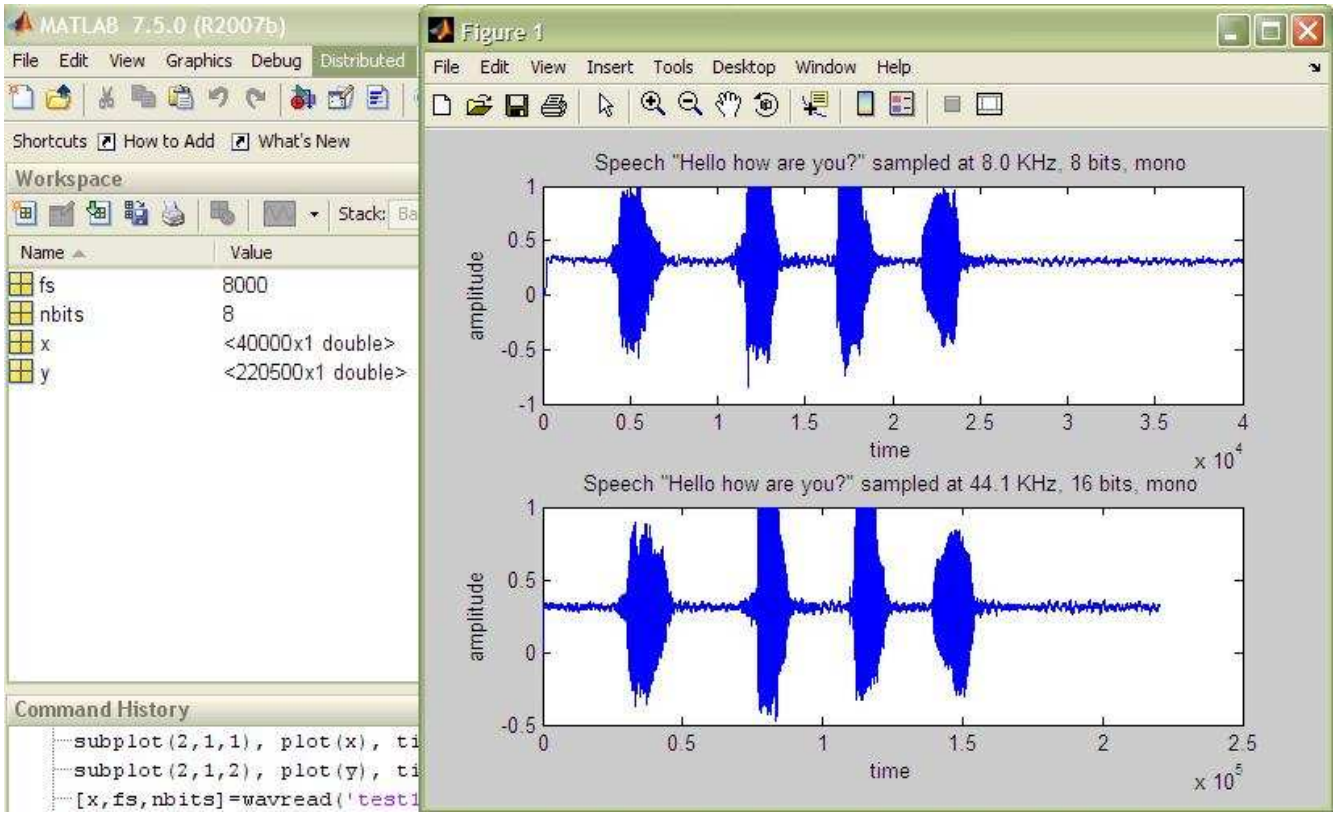

Figure 2.9 – Different samplings of the same speech signal

It might be argued that the higher the sampling frequency and the higher the sampling precision, the better the recognition accuracy. However, if a normal digital phone, which samples speech at 8.000 Hz with a 16 bit precision, is able to preserve most of the information carried by the signal [6], it does not seem necessary to increase the sampling rate beyond 8.000 Hz or the sampling precision to something higher than 16 bits. Another reason behind these settings is that commercial speech recognizers typically use comparable parameter values and achieve impressive results. So for the speech coder we decided to use 8.000 Hz of sample rate and 16 bits of resolution as configuration to record the voice.

2.3.2 Feature Extractor

After built the audioce Recorder and Player we added some new functions to the program as the part which represents the FE block. For that, we based our work on the ITU-T's G.729 Recommendation. This Recommendation contains the description of an algorithm for the coding of speech signals using Linear Prediction Coding and in which more processes like the filtering of the sampled signal, the division into blocks and the windowing of filtered signal and the word isolation are implicit. The graphic interface of the program is shown in the following Figure 2.10.

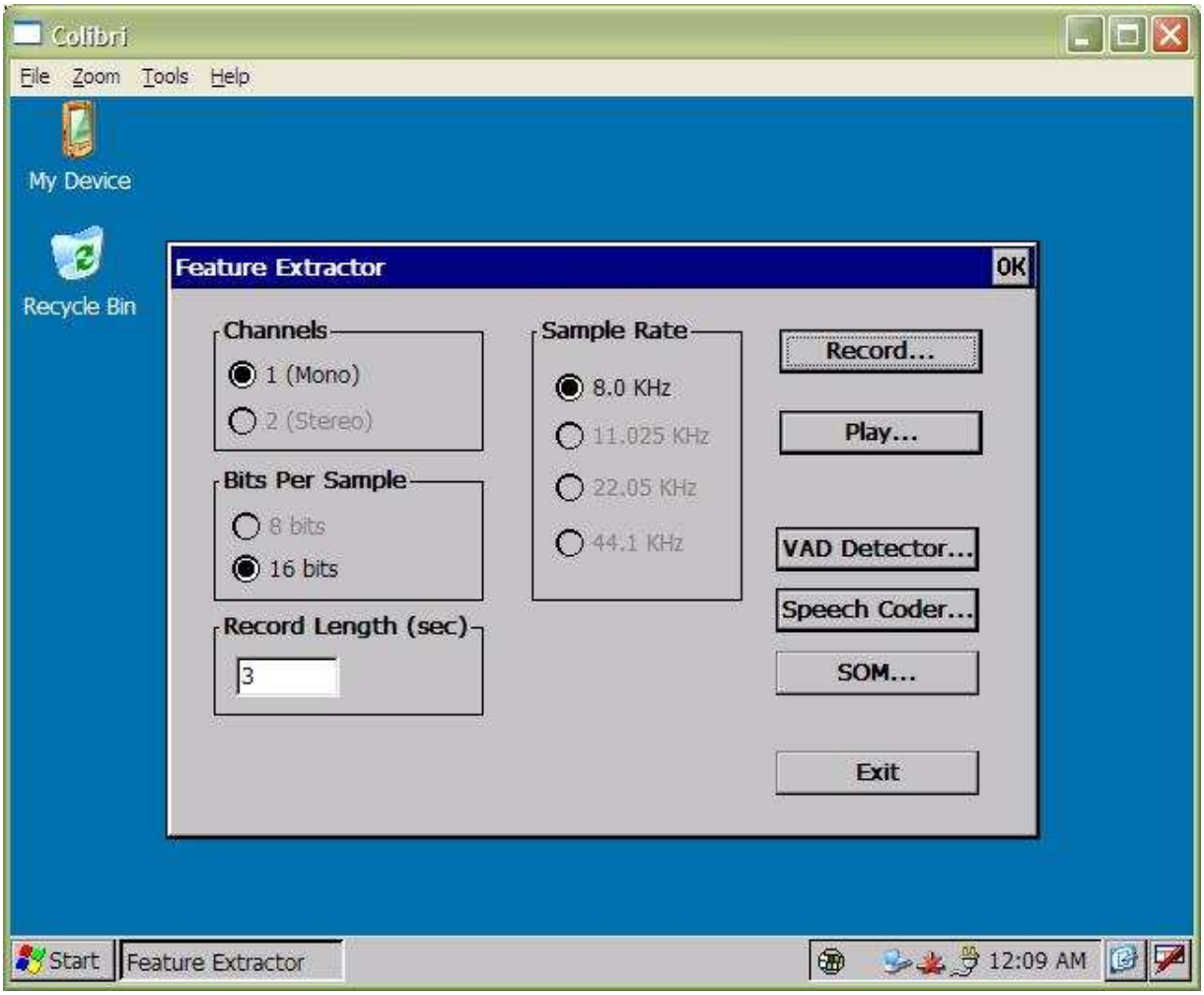

Figure 2.10 – audioce Recorder & Player & VAD Detector & Speech Coder

With this software we are able to record and play voice signals and save them in .wav files. In addition, we will be able to detect and isolate the voice command from the sound file, create a new file with it and finally codify to reduce its dimensionality. Along with the documentation (Annex 2) is included an electronic attachment containing the source code in C++ used to build the software and which is coming with all the necessary explanations.

When we press the button “VAD Detector...” we have to select a sound file which contains the recorded voice. Then the sampled signal will go through all these steps:

(Sampling: the voice signal is sampled at 8.000 Hertz with 16 bits of precision and saved in a new file called “left.wav”.)

1. Pre-emphasis filter: the sampled signal is filtered by a second order high-pass filter and saved in a new file called “filtered.wav”.

2. Word Isolation: the filtered signal is passed through the VAD block to isolate the word and saved in a file called “vad.wav”.

We can represent the results of the different stages of the signal using Matlab and the earlier created sound files (Figure 2.11).

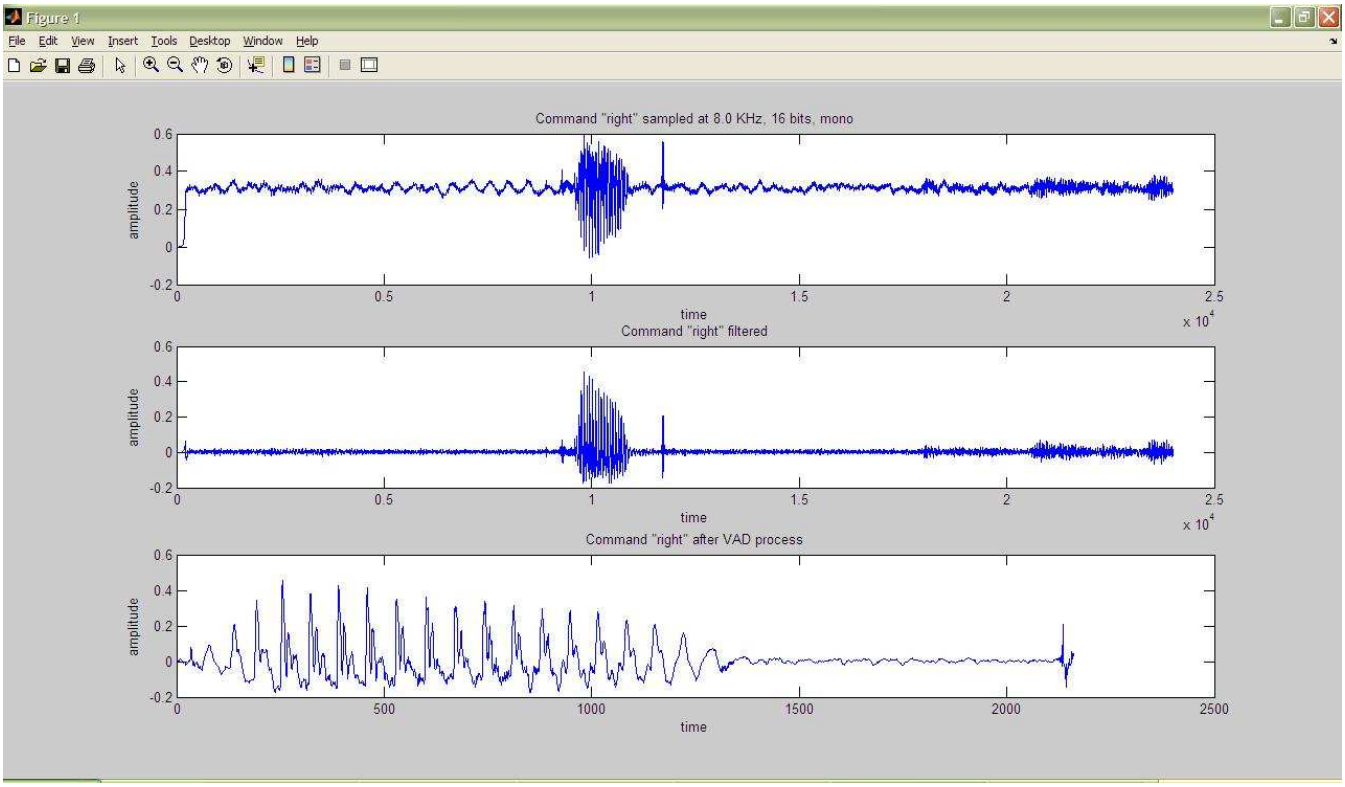

Figure 2.11 – Different stages of the signal in VAD Detector process

As we can see, the first graphic shows the original signal (Command “right”), sampled at 8.0 KHz, with 16 bits of resolution and in mono or 1 channel. We can realize that the signal has a DC offset and also that the end of the recording, is a bit noisy. The second graphic shows the filtered sampled signal and finally the third graphic shows the isolated word, after the VAD process.

At this point we have to explain that the result shown for the VAD process was obtained using an algorithm *myVAD.m* [10] obtaining pretty good results. The problem with the algorithm developed for Colibri module is that it does not avoid the silence and the non-speech parts from the voice very well as the *myVAD.m* algorithm does, as is shown in the next Figure 2.12.

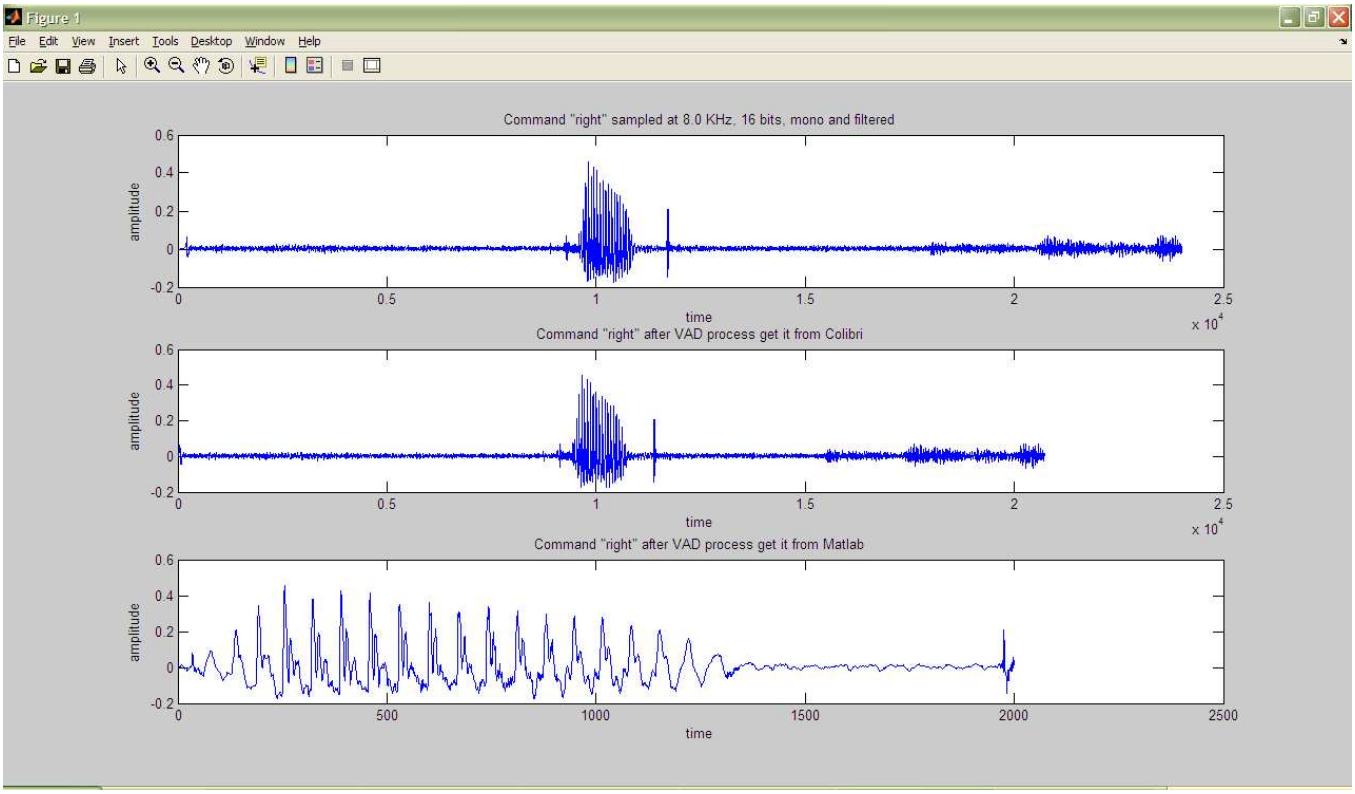

Figure 2.12 – Different ways of VAD process

In the above Figure 2.12, the second graphic shows the recorded signal after VAD process using the algorithm developed for Colibri module. Here, we can see that even though the sound file is reduced in some silent parts of the signal are avoided; these process is not doing the correct work as the third graphic does. We tried to find the solution to this but finally we did not, so this part need to be improved and is left for future researchers as we explain in the section *3.2 Directions for future research.*

To continue our research we decided to use the file created trough Matlab and *myVAD.m* algorithm called “vad2.wav”. Once we get the isolated word (“vad2.wav”) we can start codifying the speech clicking in “Speech Coder...” button. The next steps are the ones which the isolated word signal will follow:

1. Blocking: the isolated word is divided into a sequence of data blocks of fixed length, called frames and multiplied by Hamming window of same width.

2. LPC analysis: for each frame, 10 LPC coefficients are calculated.

3. Cepstrum analysis: the 10 LPC coefficients are converted in 10 Cepstral coefficient ones (Figure 2.13).

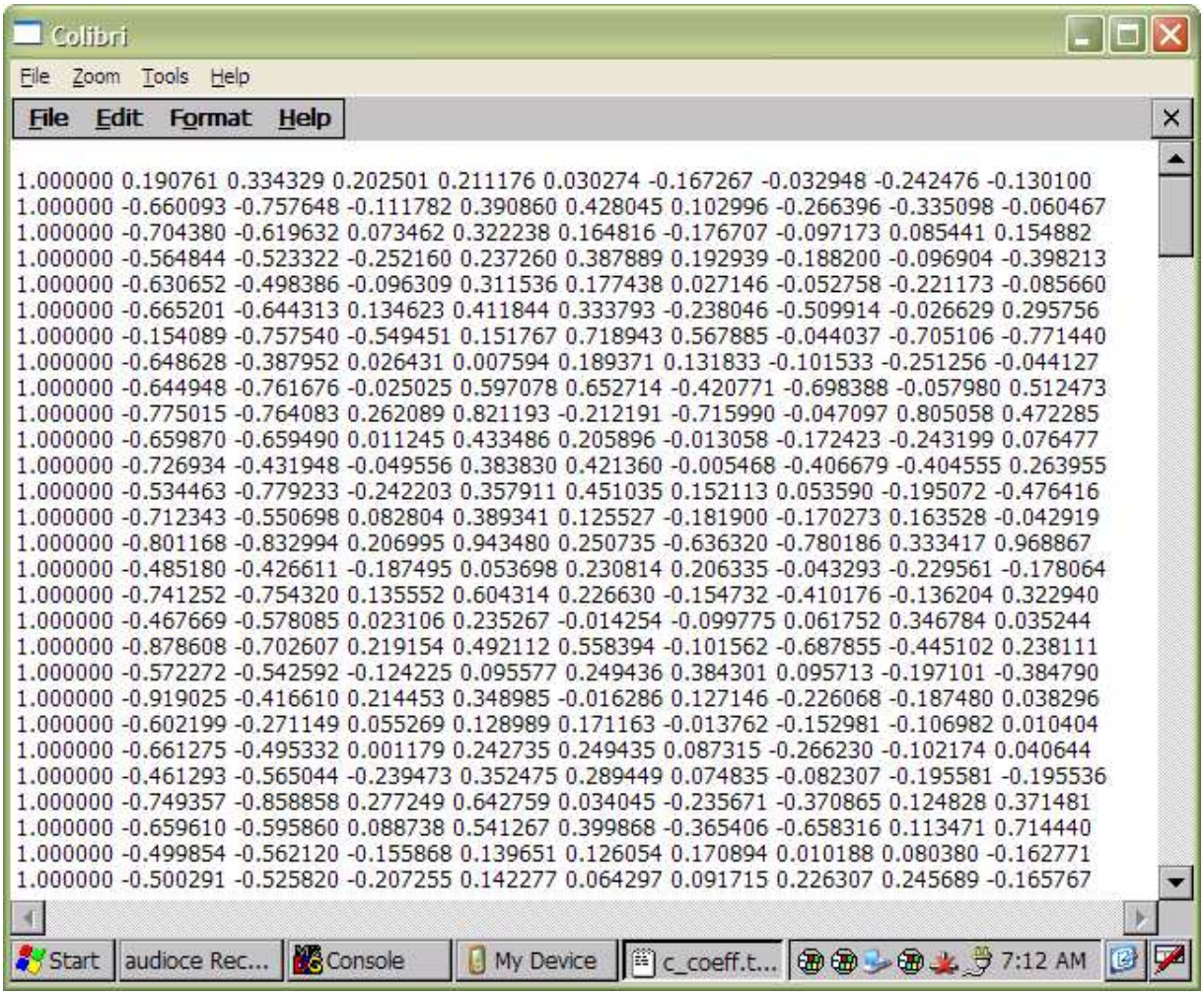

Figure 2.13 – The 10 Cepstral coefficients for each frame

Once we get the 10 Cepstral coefficients for each frame if we click in “SOM...” button and select the file with the Cepstral coefficients (“c_coeff.txt”), we will fed the input layer of the Kohonen SOM, with this input data one by one. The output layer will organize itself to represent the inputs in two-dimensional space.

The training procedure involves the following steps:

1. The neurons are arranged in an n-dimensional lattice. Each neuron stores a point in an m-dimensional space.

2. An input vector is presented to the SOM. The neurons start to compete until the one that stores the closest point to the input vector prevails. Once the dynamics of the network converge, all the neurons but the prevailing one will be inactive. The output of the SOM is defined as the co-ordinates of the prevailing neuron in the lattice.

3. A neighbourhood function is centred on the prevailing neuron of the lattice. The value of this function is one at the position of the active neuron, and decreases with the distance measured from the position of the winning neuron.

4. The points stored by all the neurons are moved towards the input vector in an amount proportional to the neighbourhood function evaluated in the position of the lattice where the neuron being modified stands.

5. Return to 2, and repeat steps 2, 3, and 4 until the average error between the input vectors and the winning neurons reduces to a small value.

After the SOM is trained, the co-ordinates of the active neuron in the lattice are used as its outputs.

Along with the documentation (Annex 2) is included an electronic attachment containing the source code in C++ used to build the software and which is coming with all the necessary explanations. In this case part of the code for SOM has been developed but still needs to be improved and finished, so this part is left for future researchers as we explain in the section *3.2 Directions for future research.*

3 CONCLUSIONS

3.1 SUMMARY OF RESULTS

As we have mentioned in the section *1.3 Contributions of this work* the main contributions of this work are the following:

- The first one was to make a market research to find a technology that would meet our needs. We found it and we also prepared it for an adequate development environment. We can see the Colibri PXA320 computer module and the Colibri Evaluation Board used for this work in the next figures:

Figure 3.1 – Colibri PXA320

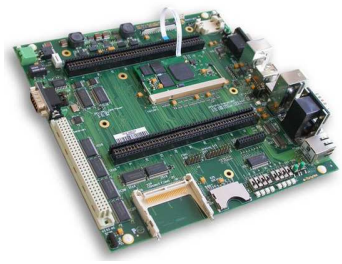

Figure 3.2 – Colibri Evaluation Board

- Then, we build a complete audio Recorder and Player. This task was not strictly necessary but the aim of this has been to learn, practice and improve the C ++ programming skills. With this audio recorder we are able to record the voice during some time, with different sample rates, different resolutions and finally we can save it in a .wav file for next processing steps like the speech coding. In addition we will be able to play and listen to the recorded signals.

- Finally, we made the FE block for the speech recognition system, based on the ITU-T's G.729 Recommendation. This Recommendation contains the description of an algorithm for the coding of speech signals using Linear Prediction Coding. This Recommendation also includes an electronic attachment containing reference C code which we used as a reference to build our own software.

The approach done in this diploma work seems to be good enough to prove that translating speech into trajectories in a feature space works for recognition purposes. The human speech is an inherently dynamical process that can be properly described as a trajectory in a certain feature space. Even more, the dimensionality reduction scheme proved to reduce the dimensionality while preserving some of the original topology of the trajectories; it preserved enough information to allow good recognition accuracy.

3.2 DIRECTIONS FOR FUTURE RESEARCH

Concerning future work, besides revising and improving the FE block the scope of the project should be to develop the Recognizer block and finally join the whole system.

Starting from the FE block the VAD block must be improved. Summarizing, we decided to use VAD (Voice Activity Detection) technique used in speech processing, instead of using the rolling average and the threshold, determined by the start and end of each word, used in the previous works, with the aim of achieving more accuracy and efficiency, but there are still some aspects in which work.

With respect to the dimensionality reduction, the source code has been developed but still needs to be improved and finished to get results. In other hand, it must take into account that as the vocabulary size grows, the reduced feature space will start to crowd with trajectories. It is important to study how this crowding effect affects the recognition accuracy when reduced space trajectories are used.

So far, all the approaches that are used in speech recognition require a significant amount of examples for each class. In a thousand-of-words vocabulary problem this would require that the user of the system uttered hundreds of thousands of examples in order to train the system. New approaches must be developed such that the information acquired by one module can be used to train other modules i.e. that use previously learned information to deduce the trajectories that correspond to non-uttered words.

Finally, the scope would be to join the two parts of the whole project. The first part is the control module for the management of the wheelchair and the system of an ultrasound sensor net

to capture data from the surroundings, such as location and distance barriers and the second one is speech recognition system to control the wheelchair. Also the scope would be to convert the whole system in a real time system, instead to continue being a simulation as has been until nowadays.

4 REFERENCES

[1] G. Pačnik, K. Benkič and B. Brečko, *Voice Operating Intelligent Wheelchair – VOIC*, Faculty of Electrical Engineering and Computer Science, Institute of Robotics, Maribor, Slovenia.

[2] Pablo Zegers, *Speech recognition using neural networks*, University of Arizona, Arizona, 1998.

[3] Sung-Won Park, *Chapter 7: Linear Predictive Speech Processing*, Texas A&M University-Kingsville, Texas, 2007.

[4] Teuvo Kohonen, *Self-Organizing Maps*, Springer-Verlag, Berlin, 1984.

[5] ITU-T Recommendation G.729 Annex B (2007), *A silence compression scheme for G.729 optimized for terminals conforming to ITU-T Recommendation V.70.*

[6] Laurence Rabiner, Biing-Hwang Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, 1993.

[7] R. M. Gray, *Vector Quantization*, IEEE ASSP Magazine, April 1984.

[8] K-F Lee, H-W Hon and R. Reddy, *An Overview of the SPHINX Speech Recognition System*, IEEE Transactions on Acoustic, Speech and Signal Processing, vol. 38, no. 1, January 1990.

[9] Toradex AG, Privately held shareholder company, http://www.toradex.com/En/About, Switzerland, November 2003.

[10] Olutope Foluso Omogbenigun, *Speech Core - A complete bundle of m files for Isolated Word Speech Recognition*, http://www.mathworks.fr/matlabcentral/fileexchange/19298, London Metropolitan University, 2009.

EPILOGUE:

ANNEX 1: Student address and Short Curriculum Vitae.

ANNEX 2: DVD with all the documentation and software.

ANNEX 1:

Student address

Eneko Añorga

Principova ulica 9, 2000 Maribor

Tel.: 070/266-062

E-mail: enekux@gmail.com

Short Curriculum Vitae

Date of Birth: 18.07.1983

Education:

2001/2006:

Telecommunications Technical Engineering, Major in Telecommunications systems, University of Mondragon.

2006/2008:

Electronic Engineering, ETSETB - UPC (Polytechnic University of Catalonia).

2008/2009:

Diploma Work, FERI, Univerza v Mariboru

ANNEX 2:

DECLARATION:

I, Eneko Añorga, the undersigned, declare that I have made the diploma work by myself. I am aware of the potential consequences in the event of a breach of this declaration**.**