CHALMERS

A Voice Controlled UHF Radio

DANIEL HERNÁNDEZ TIELAS

Department of Computer Science and Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden

A Voice Controlled UHF Radio

Theoretical study and implementation of current voice recognition algorithms in Java for the remote control of a UHF radio

DANIEL HERNANDEZ TIELAS

ISBN

Examiner: Lars Bengtsson

Daniel Hernandez Tielas Civic Number : 890617-C536

Department of Computer Science and Engineering

Chalmers University of Technology

tielas@student.chalmers.se

Telephone: +46 760652163

Printed by Chalmers Reproservice

Acknowledgement

Develop a final master thesis not only implies a huge workload, it is also the work that concludes a five-year project. All the efforts and sacrifices are rewarded at the end, but as all the rewards, it carries thanks to the people who have made it possible.

First of all I would like to thank to my supervisor Lars Bengtsson who has guided me in this project. He gave me the idea and he has been advising me with comprehension and respect.

Since this thesis has been developing in this year abroad, in Sweden, I would like to thank to my Erasmus family that help me to overcome all the difficulties in this intense, amazing and short year. Specially mention to Nuria, Eli, Remi, Teresa, Marcio, Cesc, Damian and Francesca who helped me with the recordings of the commands.

Finally, I would like to thank all the support, love and help received from my family. Without them, none of this could have been possible.

Abstract

A Voice Controlled UHF Radio

Department of Computer Science. Chalmers University of Techonlogy

Hernández Tielas, Daniel

Nowadays, more and more devices attempt to improve user interaction including the use of speech recognition. This project responds to the need of a company for developing an efficient system for voice control of an operative UHF radio in sporting environments.

The work, conducted at Chalmers has been divided into two main parts. First, a theoretical study and analysis of the "state of art" in voice control algorithms has been performed. The initial objective of this part was to familiarize and understand how diverse systems and algorithms in voice recognition work, even testing and modifying some of them.
In the second phase, the thesis has been focused on developing and testing a demonstrator device with the concepts and ideas learned from the theory. An Android device recognizes specific voice commands and sends the orders to the UHF radio by a combination of serial and TCP/IP communications. For control with maximum accuracy and minimum power consumption, Hidden Markov Models (HMMs) and Dynamic Time Warping (DTW) algorithms for voice recognition have been studied, optimized and implemented through several applications in Java.

The thesis concludes with several reflections about how the thesis has been developed and it introduces some ideas and recommendation to improve the system in future researches.

Key words: Voice Control, UFH radio, demonstrator development and testing, efficiency Dynamic Time Warping (DTW) algorithm, Hiden Markov Models, Java, Android.

Table of Contents

Chapter 1

Introduction

Simple ways to control devices are directly related to natural human ways of communication. In this sense, it is highly desirable to be able to control any device in such a natural an intuitive form as by the voice, especially while playing sports or doing different activities the hands are not free to manage any device.
Develop an accurate voice control system is not trivial; there are many different implicated issues, and therefore to success it is necessary wide knowledge in a many different fields as phonetics computation, signal processing, communications, or control algorithms. Much of the complexity of these systems is directly related to the inherent complexity of language. A language is a complex system where many concepts and variables are involved and interconnected; any language is structured by a combination of phonetics, phonology, morphology, syntax, pragmatics or semantics.

1.1. Acknowledge in language processing

A whole human language is a complex system determined by multitudes of different variables and parameters. Without one of them, it could be impossible to understand a sentence, a word or even a basic sound. In this subsection, there will not be any depth study about the language, but it is necessary to present briefly all the parts in the study of a language. As Chapter 2 of reference [1] reflects any language is composed by:

• Syntax: it is related to the study of the principles and rules for constructing phrases and sentences. The meaning of a sentence can change or be incomprehensible if the speech recognition model does not process correctly the sentence.

• Semantics: it is referred to the meaning of the words. Lexical and compositional semantics; the meaning of a word can vary depending on the context, the composition around the word. (See ambiguity)

• Phonetics and Phonology: they are the science which studies the sound of the human speed and how it is done. Phonetics is concerned with the physical properties (acoustic properties, neurophysiologic status and physiological production). On the other hand, Phonology is related to the grammatical characterization of systems of sounds or signs.
Both of them have particular relevance in the speech recognition process. It is necessary to know how a word is pronounced, and its physical properties to detect and recognize appropriated sounds.

• Morphology: is the study of the meaningful components of words. It includes identification, analysis and description of the structure of a given language's morphemes and other linguistic units like affixes, intonation, stress or implied context.

• Pragmatics: it studies the ways in which context contributes to understand the meaning. It is relevant when the voice works with complex orders in different contexts.

• Discourse: referred to the knowledge about linguistic units larger than a single utterance.

In completed speech recognition systems where ambiguity could be a problem to determine accurately the meaning of sentences, uncertainties can be solve using what is called word sense disambiguation. However, grammatical structures for our voice control system are simplex, so in this project this possibility will not be considered.

1.2. Introduction to Models and algorithms

The research and development of models and algorithms for speech processing has experimented a breakthrough in the last 60 years. The current knowledge involves no so many formal models or theories, but it represents the culmination of the collaboration between distinguished scientists and mathematicians, almost always linked to university research in fields as varied as natural language in computer science, speech recognition in electrical engineering, or computational linguistics and psycholinguistics.

Among the main models, it is possible to find finite-state machines, rule systems, probabilistic, logic, and vector-space models. In turn, the numbers of algorithms constituents of the models are no so large, emphasizing among them the state space search algorithms and machine learning algorithms.

The State machine model is a formal mathematical model. It is thought like an abstract machine which can be only in one state at the same time (current state). However, it can change from one state to another with the appropriated event or condition. (Transitions and input representation). Deterministic and non-deterministic, finite-state automata and finite-state transducer are considered variations of the basic model.

Regarding formal rule systems, the most influential models are regular grammars and regular relations, featured-augmented grammars and context-fee grammars considering also their probabilistic relations. When the study is focused on phonology, morphology and syntax, formal rule systems and State machine are specially recommended.

Logic is also important in language processing. In this field, first order logic-more known as predicate calculus is widely used with some concepts about formalisms like lambda-calculus, feature-structures, and semantic primitives. In the past, the logic has been applied for modeling semantics and pragmatics, but in current works it has focused on robust techniques drawn from nonlogical lexical semantics.

On the other hand, probabilistic models cannot be obviated; Probabilities can augment the three presented models (state machines, formal rule systems, and logic). If the state machine is augmented with probabilities, then it becomes the weighted automaton or Markov model. Hidden Markov models or HMMs are extremely useful for multiple tasks in this area, for example, it is going to appear in text-to-speech, in speech recognition or machine translation. These models are extremely significant because they permit to solve different types of problems with ambiguity; if there is some ambiguity in the input, they permit to choose the most probable model.

Graph algorithms, as depth-first search, are quite common when nonprobabilistic tasks, as machine state, are faced. For probabilistic tasks, heuristic variants like best-first or A* search are the most usual methods. Decision trees, logistic regression or Gaussian Mixture Models could be other essential tools presented in voice recognition.

Chapter 2

State of Art

Nowadays, the voice control of different devices is achieved in an acceptable way in both precision and efficiency, always if the input fulfills the quality requirements.

Technology marked is highly competitive and it is always asking for continues improvements which permits to gain marked spread. High investments and deep researches have been done in the last twenty years, especially within the computer and telecommunication industry. These investigations and developments have been mainly focused on two main applications: Continuous Recognition Speech and Isolated Voice Recognition.

Although this paper is orientated to isolated voice recognition, software for continuous speech recognition is also described since continuous systems are large extensions of the isolated word recognition processes.

2.1. Voice Control Applications

2.1.1. Continuous speech recognition. Speech to text Systems

This technology translates spoken words into text. It can be used just to convert speech into text or this functionality can be extended: once the text is gotten the system can analyze the text and acts in consideration of its content. These are complex systems that not only have to deal with words recognition; they have also to consider the sense of these words when they form sentences, grammatical structures must be correct, and whereby the system vocabulary and complexity increases significantly. Speech recognition applications cover voice user interfaces like analysis of telephone conversation, speech-to-text processing for email or searches in voice databases. The most significant software related with this application is next presented. The software description is updated to June 2012.

Windows

Windows has a speech recognition system called Windows Speech Recognition and it is integrated in Windows Vista and Windows 7 which includes version 8.0 of the Microsoft speech recognition engine. This system can be improved with some add-ons for Windows 7 as: VoiceFinger, WSR, Tooljit, Trigamtech or Vocola.

Other relevant software running in Windows is:

- *Vox Commando*: It permits the control of media programs which run in Vista and Windows 7 such as: Itunes, XBMC, Skype and more.
- *E-Speaking and Tazti:* They are focused on the control of almost any desktop software by voice. It allows custom speech commands.
- *Sonic Extractor* (Specialized in Telephony and broadcasting).
- *Dragon Natural Speaking:* Focus on dictate and speech transcription.
- *SpeechMagic of Philips:* Focus on Medical industry.

Macintosh

- *MacSpeechScribe:* Focus on transcript recorded dictation into text.
- *DragonDictate*: Dragon Version for Mac.
- *iListen*: A third party software which allows text input using the voice.
- *Plain Talk:* Apple´s speech synthesis and recognition system

Open source

- *CMU Sphinx:* By Carnegie Mellon University. It offers several speech recognizers, an acoustic model trainer, a language model compilation and an open pronunciation dictionary. It is a completed set which has been used in this thesis to learn the voice control principles and its basis. It has been useful to get a general idea of the whole process operation and the general structure of the system. A free version of this software was firstly adapted to gain knowledge and have an auxiliary and preliminary solid version. The package comes with extend and well explained documentation and tutorial which has been deeply studied. They have taken as an important reference in this work. See Reference [2]

- *ISIP software:* It is software developed by a research group in the Electrical and Computer Engineering department at Mississippi State University. They have created a freely available, modular, state-of-the-art speech recognition system. This system can be easily modified to suit other research needs.

- *RWTH ASR:* It is the RWTH Aachen University open source speech recognition toolkit.

- *Sclite:* It is open software to score and evaluate the output of speech recognition systems. This tool is inside of the NIST SCTK Scoring Toolkit developed in Berkeley University.

·*VoxForge:* Is a free speech corpus and acoustic model repository for open source speech recognition engines.

Software for Smart phones and embedded systems:

A lot mobile phone and other mobile devices are dated with some type of speech recognition system in order to facilitate tasks as different as Internet Browsing, texting, playing games or just dialing. It is a very attractive field for users who cannot control the device with hands in many situations. A high number of third party applications have implemented language speech support.

Ordered by OS:

Android:	**Apple's iOS**	**Multiple OS**
·*Ziri Assistant Beta*	·*Dragon Dictation & Search*	·*Google Voice*
·*Speaktoit Assistant*	·*Siri Personal Assistant*	·*Bing*

Embedded Systems:

·*ViaVoice:* IBM continuous speech recognition software for embedded systems.

·*Voicelib:* It is a modular C-language implementation of voice recognition *algorithms*. It allows fast prototyping of user interfaces and interaction styles.

·*Sensory's FluentSoft:* Advance technology in speech recognition focus on highly accurate with low-cost. The software can be applied inter alia to feature voice recognition and synthesis in text-to-speech, interactive robotic controls or MIDI-like music.

·*VoiceBox Embedded Solutions:* Simplifies the use of Telematics, Navigation and Infotainment through embedded platforms allowing Free-form voice commands, Voice Search and control of Satellite Radio, HVAC and other ECU devices.

2.1.2. Isolated words recognition. Voice Command System

This type of voice control is characterized by single command recognition. A command may be composed of one or more words, but they are always predetermined. The system must listen to the hypothetical command and compare it with a set of references; then if the sound is similar enough to any reference according to voice control algorithms, the command will be accepted as valid, if not it will be rejected.

The system uses a limited number of acceptable commands, so system complexity is significantly lower. This system gets higher accuracy, with fewer resources, but the possibilities of the system are dramatically reduced. Most systems are designed to recognize specific words within loudly speeches or noises. The main applications of this system are in robotics, home automation (Domotics) and Telematics (e.g. Vehicle Navigation Systems).
Available software for this application is mainly developed by universities as a way to teach and train students improving the knowledge in this area. Significant examples are the *Signal Modeling for High-Performance Robust Isolated Word Recognition* at Old Dominion University

or the *Time-Delay Neural Network Architecture for Isolated Word Recognition* at Carnegie-Mellon University.

Professional software is short, unknown and often restricted to each specific device. Adaptation of more complex and commercial software is a common practice. It was this lack of commercial software which reasons this theoretical study of algorithms for voice control, and its subsequent practical implementation.

2.1.3. Other Utilities

Besides the commercial PC and smart phones, on the market it is possible to find many other devices that use the user's voice to be controlled. Today there are many more examples than lines in this publication. Only few samples are mentioned:

•Mostly all the commercial GPS devices in the marked have implemented the voice interaction system with the terminal. This allows the user the device to control in hands-busy or eyes-busy mode.

•In the automobile sector, nowadays almost all the upper class cars have a central computer. The users can control functions like the music, air conditioning, and route or even speed only their voices.

•The use of Dictation is a reality in areas such as law and augmentative communication. The communication by voice with the computer is especially beneficial among people with some disability unable to type.

•Publisher as Pearson or Educators use testing services like ETS or automated systems to analyze student essays, grading and assessing them, when it is an impossible task for their human staff.

•Text analysis companies are working efficiently using voice recognition systems, providing marketing intelligence based on automated measurements of user preference and opinions. Speaker recognition systems are widely used in this area.

Despite this technology seems to be fully developed, there are still numerous challenges to overcome. For example, a big challenge for voice or sound control is to detect, analyze and act according not only according to the meaning of the message; also taking care of extra parameters like intonation, speed or tone of the message. That can be extremely useful in order to find the best solution for each client request in each personal situation. Solutions for simultaneous conversations or detection of problems in environments or machines by listening and recognizing noises will be other fascinating applications of this technology in the close future.

Chapter 3

Basis of Voice Control

The control of devices by voice is a complex art with a rapid evolution. Today it is possible to find devices able to analyze and recognize complex discourse with some precise tolerance. Regardless, this work does not go so far; it is focused on command recognition in sports, which reduces the number of words to recognize but increases the chances of failure by environmental noise. The basis on automatic speech recognition (ASR) from the input signal processing until the final matching is described in this chapter.

3.1. ASR: How it works

The goal of Automatic Speech Recognition (ASR) research is to address this problem computationally by building systems that map from an acoustic signal to a string of words. It differs from Automatic Speech Understanding since ASU extends the goals to produce some sort of understanding of the sentence, rather than just words.

A successful development can only be achieved with prior good design, and the first step to get an excellent design is start thinking about goals and requirements of the product. It is indispensable to define and the type of costumer to whom it is orientated, and the needs that the product should cover.

Once answers for all these questions are defined, it is feasible to ask with what kind of parameters the model is desirable to work; there are four dimensions or parameters of the ASR system that should be considered.

• How natural, how fluent is the speech that the system has to recognize. It is different if we are using a specific isolated word, if we are using a string of words with long pauses between them,

or if the speech is continuous and fast, for example, in a business dialogue. The system of this work is ready to understand commands within a normal conversation where words are not overlapped and when they are well pronounced in a soft back noisy environment.

•The vocabulary size that the system has to manage. The complexity of the speech recognition is higher with the number of words that it has to operate. It is not the same to work with a huge number of words like in transcribing human-human conversation with more 64000 words, than work with a reduced number of words like recognizing sequences of digits or answers like yes or no to a question. The designed system works with a vocabulary of 9 words, which can be combined to form a total of 15 commands.

•The channel and noise make the recognition harder and they obligate the system to uses different filters and extra signal processing. The working system is ready to work in a soft-noisy environment. Wind background noise or soft dialogs will not introduce disturbances in system operation.

•Variation in accent or specific language characteristics difficult the recognition. A clear and well pronounced sound will make the task easier, but in general the specified system should be able to recognize commands from any speaker.

3.1.1. Voice Command Recognition Architecture

The objective of a speed recognition system is to get the correct written message in the output, taking as input an acoustic waveform which is mixed with noise.

A typical system for Voice Command Recognition presents the topology presented in figure 6. The basic idea of the process is progressively record every sound that could be a command, extract the features of this sound and finally compare these features with the features of the library commands using different algorithms. If the features are similar enough, that is to say, if the reference and the data match, then the word can be accepted.

The process explanation is extended step by step:

1. The system listens and frames the input signal. Typically a sampled rate of 8000Hz is used. (This is the predetermined rated offered by the phone). Since it is interesting to get all the variations of the sound, each frame will have 256 samples and therefore one frame is gotten each 8000/256= 32ms. This period will be enough to catch any significant fluctuation. No overlap is programmed.

2. The energy of each frame is calculated applying equation VII. If the frame has enough energy (above a prefixed boundary), then the frame is recorded. The flag with binary values is useful to consider whether the previous frame had high energy or not, in order to keep recording in the same file or not. This flag and some other extra variables allow the system to get different recording for each potential word.

3. If the consecutive set of frames with high energy has enough frames to form a potential pronounced command, this record will be analyzed.

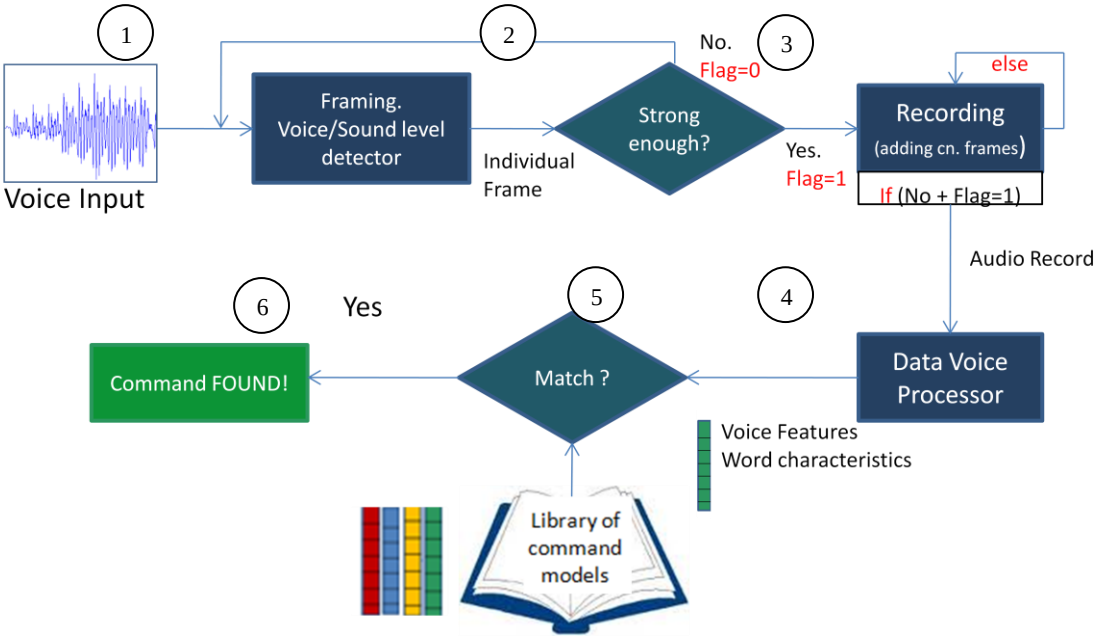


Figure 1: Command Recognition System Architecture.

4. These recordings are processed in the data voice processor in order to convert the audio data into a set of representative vectors called feature vectors. For each sample of the record the system will obtain a feature vector of 13 or 39 MFCC components, depending if the 26 delta coefficients are considered or not. The final system developed for the thesis works with 13 components because as it is reflected in the appendix, working with those 39 elements does not introduce a significant improvement in the accuracy and it increases in three times the calculations.
   At the end of the extraction process, each sound will be represented by a matrix of dimensions $Number\_frames \times 13$.

5. The matrix of characteristics of the current sound is matched with reference commands matrix in the library. This process is called features matching and the study and development of the required algorithms for this task are the main objectives of this work.

6. If the pronounced word fits with a word from the library then the word has been recognized and the subsequent message will be send to the radio communication system to complete the action required by the user.

It is important to remark that all the audio recordings in this work have been done with a Samsung Galaxy S Plus, with the same sample rate 8000Hz and similar environmental conditions. This method considers the input acoustic waveform as a noisy signal where the pure

message is distorted. Therefore, the basic approach to avoid a deep study of the noisy channel is to make all the recording (training, testing and working) from the same terminal. In this way, it is possible to ensure same distortion for all the uses and no extra actions need to be taken.

But even if the same channel is constantly used, the speech is always variable; it is impossible to get two identical pronunciations of the same word. Therefore, the matrix of features from the library and the audio candidate matrix will not ever be exactly equals. There are many parameters which make these signals different such as speech speed, pronunciation, duration, tone, or intensity. Several algorithms are developed to make the comparison feasible.
After the general review of system architecture, two main problems for the system stand out from it:

·Convert the audio input into a set of characteristics which define the sound.

·Determine if the extracted features corresponds to the pronunciation of a key_word.

3.2 Voice Feature Extraction and Vector Quantization

3.1.2. Voice Feature Extraction (MFCC)

An optimal extraction of the parametric representation of an acoustic signal is essential for the recognition task since this parametric representation will be later utilized in the features matching process. If the input signal is not well represented by the parametric matrix, then any successful recognition will be impossible.

As it is shown in Figure 2, MFCC process is composed by seven computational steps.

Each step or phase is briefly discussed in the following page. It is worth realizing that the first two steps are automatically done by the Samsung Terminal, but it is important to describe them in order to have a suitably understanding of the whole process.

1. Pre-Emphasis

It consists of emphasize the higher frequencies of the signal passing it through a specific filter which follows the structure:

$$Y[n] = X[n] + a \times X[n-1] \qquad \text{(I)}$$

Each *"a"* has determined value which depends on subjective criteria about the continuity of the samples. For example, a=0.90 means that 90% of any sample is considered to be generated from a previous one. The exact value or procedure followed by the phone manufacturer is of course unknown.

2. Framing

The speech signal varies slowly in the time domain. If it is analyzed in a short period (between 5-100 msec) its characteristics barely change. However in longer periods between 0.25 to 0.4 seconds its characteristics change reflecting different pronounced sounds. This is why speech

frames obtained in the recording process are segmented into frames of 0.32 seconds with an optional overlap which is not considered for simplicity. In this way, it is nearly possible to adapt each frame with each phoneme.

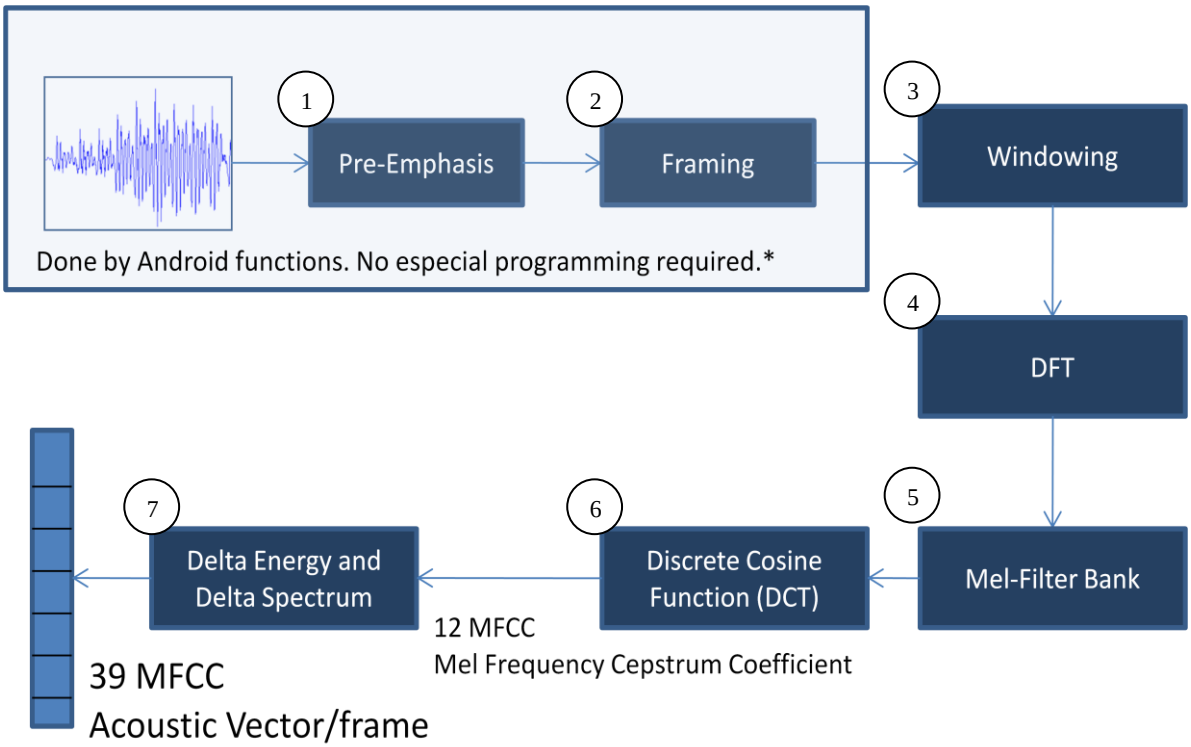


Figure 2: MFCC Process

3. Windowing

In order to maintain the continuity at the first and the last point of a frame, it is necessary to multiply each frame by a hamming window. This window shape considers the next block in feature extraction, and it integrates the closest frequency lines. After multiplying each frame by a Hamming window, it is possible to consider the speech signal periodic and continuous. As it is probed in [3] Hamming Window gives better performance than triangular or rectangular windows when linear and Mel scale are contemplated.

Windowing operation

Input signal $X(n)$ *with n=0....N-1 N= Nº of samples per frame*

Hamming window $$W(n) = 0.54 - 0.46 \times Cos\left(\frac{2\pi n}{n-1}\right) \quad \text{(II)}$$

Output signal $$Y(n) = X(n) \times W(n) \quad \text{(III)}$$

4. Fast Fourier Transform

The variability in time domain of the speech signal is significantly high to process the signal in time domain.
It is probed that different tones corresponds to different distributions of energy over the frequency, therefore applying FFT to get the magnitude frequency response of each frame will be particularly useful to identify different tones.

$$Y(w) = FFT\ [H(t) \times X(t)] = H(w) \times X(w) \qquad \text{(IV)}$$

5. Mel-frequency Wrapping and Mel-Filter Bank

The Mel-Cepstrum technique proposed by S.B.Davis and P.Mermelstein [4] tries to eliminate the contribution of the model to the filter through an integration model in the frequency-domain. To do this, first compute the periodogram of the signal section, squared modulus of the DFT of the section. Subsequently, over this spectrum is applied a bank of triangular band-pass filters distributed uniformly on the mel-scale range over desired frequencies. The spacing as well as the bandwidth is determined by a constant mel-frequency interval depending on the sampling rate.

$$F(Mel) = 2595 \times \log_{10}\left(1 + \frac{f}{700}\right) \qquad \text{(V)}$$

Note that these filters can be represented as matrixes, so any filter could be applied to DFT bin output by matrix multiplication.

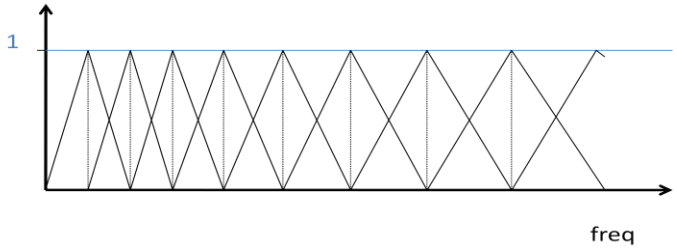


Figure3: Bank-Filter in Mel scale

The values, on a logarithmic scale, resulting from integrating the weighted averages of the periodogram in each band of frequencies, are used to compute the mel-Cepstrum coefficients thought the step 6, the Discrete Cosine Transform.

As a result of this process, some coefficients are obtained which take into account the logarithmic sensitivity in intensity and frequency of the human ear and its resolution in critical bands. Each Mel filter precisely corresponds to a critical band, approximately 20 within a range of frequencies of 4 kHz. With this procedure the amount of data is reduced; instead of having a number of values as bins produced by DFT, the system will count with as many values as number of filters.

6. Discrete Cosine Transform (DCT)

Log Mel spectrum is converted into a time-like domain called *quefrency* domain by applying DCT. The values obtained after this step are known as Mel Frequency Cepstrum Coefficient. The completed set of coefficients is a vector called acoustic vector, and it usually has 12 components (K=12)

The formula used to compute DCT is shown next:

$$C_n = \sum_{k=1}^{K} (\log S_k) \times \left[n \times \left(k - \frac{1}{2} \right) \times \frac{\pi}{K} \right] \qquad \textit{where } n=1,2,....K \qquad \text{(VI)}$$

Computing Discrete Cosine Transform of log filter-bank energies allows getting uncorrelated MFCC coefficients.

7. Extraction of energy and Delta Coefficients

The energy of the frame is also a feature of an acoustic signal and it can be easily computed. It will mainly help the system to differentiate when there is an energetic input or just soft noise. The energy of a frame represent by the acoustic vector X of dimension 12 can be calculated as.

$$Energy = \sum_{i=0}^{i=11} X^2[i] \qquad \text{(VII)}$$

If after all the previous process the energy coefficient is added to the acoustic vector, then the system will have converted each analog audio 32ms fragment into an acoustic vector of 13 elements. However within these coefficients it is not possible to find anyone that represents how the voice signal and the frames changes, so for specific applications it is recommendable to add what is denominated Delta coefficients to reflect these changes.

The delta coefficients represent times derivates of the acoustic vector. The first 13 delta coefficients show the velocity and the second 13 delta the acceleration. The equations to compute these coefficients are:

Speed $$d_1(t) = \frac{c(t+1) - c(t-1)}{2} \qquad \text{(VIII)}$$

With d(t) as the delta coefficient at time t and c(t-1) as the acoustic vector coefficient at time t-1

Acceleration $$d_2(t) = \frac{d1(t+1) - d1(t-1)}{2} \qquad \text{(IX)}$$

If t=0 → $d(t) = c_{t+1} - c_t$ *If t=T* → $d(t) = c_t - c_{t-1}$

3.1.3. Vector Quantization (VQ)

As result of the Voice Feature Extraction process, an analog speech audio of Nx32msec has been converted in a matrix of $(N \times 32) \times 13$ doubles which represents the characteristics of that audio.

An important issue to consider in any kind of voice processor is the speed and efficiency that the system is working with. Since human language is built by a predefined and fix set of sounds or phonemes, if different observation or acoustic vectors from a speech are analyze it is possible to find that some vectors are close from each other; they represent the same phoneme.

That fact allows applying the VQ technique to classify each acoustic vector according to the sound that it represents, by substituting this vector for a number (best representative vector) that identifies the kind of vector (phoneme) that is at stake.

The best way to measure the accuracy of any quantifier is to determine the error that it produces when the conversion takes place. This parameter is called error by distortion. The aim of a quantifier is to obtain the best representative acoustic vectors from all the acoustic vectors which represent the same phoneme getting the minimum error by distortion. All the representative vectors (codewords) form a Codebook. In this way when any phoneme is pronounced, the subsequent acoustic vector should be close enough to the correct representative centroid in order to classify properly that phoneme. Similar or close vectors are classified in the same region or cluster and they are represented by the representative acoustic vector called Centroid. See figure 4.

The advantages of this technique are multiple since it reduces the storage of information and makes further processing faster and more efficient. Each observation or acoustic vector will be represented by an integer number, and therefore instead of using a $(N \times 32) \times 13$ matrix of doubles to represent an acoustic input of $N \times 32$ msec, it will use just a vector of N integer components.

The disadvantages associated with this technique came from the difficulties to establish a correct codebook which minimize the distortion error, and the subsequent computational work that it requires. A small codebook keeps distortion error, but a too large codebook could lead to get different centroids for the same phonemes which would drive the system to future confusions. After previous reviews of studies in English Phonetics (see Chapter 2 in reference [1]) and after analyzing the phonemes and noises that compounds the commands of the control system; it has been decided to configure a codebook with 64 centroids from the recording of 5 different speakers. The outcome is optimal for this election. The result of the work with 16, 32 and 64 centroids for only one speaker is summarized in Table 6.

Components of a vector quantizer

To build a vector quantizer it is required:

1. A big set of observation vectors resulting from the signal processing of each word repetition. This training set should be as bigger and diverse as possible in order to classify and quantify all the further possible phonemes. It was used a set of 1620 observations as the result of processing records from six different speakers.

2. A measurement of the distortion in the classification. This measurement will be given by the addition of all the Euclidian distance between each observation vector and its assigned centroid in the codebook.
3. A procedure to place and recalculate the centroids according to the minimum distortion requirement.

Euclidian distance

This measure is referred to the distance in a straight line between two points in a Euclidean Space. In VQ, this distance is useful to measure the distortion error of the Codebook. In a Space of 13 dimensions, the Euclidean distance between two vectors ($V_{1,}$ V_2) or points is given as:

$$Euclidean.d = \sqrt{\sum_{i=1}^{i=13}\left(V_{1,i}^2 - V_{2,i}^2\right)} \qquad (X)$$

It was found that this measure of distance requires less processing time than other distances since this one requires simple calculations. In some voice recognition literature, Mahalanobis distance is recommended to measure distances or distortions. However, this last technique requires calculation of the covariance matrix inverse which means a greater number of operations and processing time.

Vector Classification

The aim of the VQ process is to classify a big number N of observation feature vectors into a small quantity of groups M (M<N) where similar vectors (sounds) share the same group.

There are many criteria to get acceptable classifications but only the most relevant algorithms are presented in this paper. The main classification algorithms are: K-Means and the LBG algorithm.

The N original Vectors will be represented by M vectors (centroids or codewords) which defined M different space regions. All the codewords constitute a Codebook.

K-Means Algorithm

This classification method is called K-Means because the vectors are grouped into k intermediated values, resulting *k* final sectors or spaces *(k = M).* The main problems are the correct selection of the initial values μi, and the estimation progresses algorithm. The followed process is presented below. Pseudo code in Appendix B could help to understand the specified process.

1. Initialization: choose arbitrarily M vectors or codewords as the initial set of codebook.

There are some noteworthy techniques which avoid arbitrariness to get an advantageous initial set. The used technique takes the vectors with maximum and minimum coordinates to set the initial Codebook as it is indicated:

$$Coordenates\_c_n[i] = \left(\left(\frac{MaxCoordenates[i] - MinCoordenates[i]}{cluster.lenght + 1} \times n \right) + MinCoordenates[i] \right)$$

2. Finding the nearest: For each observation vector, finds the closest codeword in the codebook (in Euclidean distance), and assign that vector to the concerned cell.
3. Centroid update: Once that all the observation vectors belong to a specific region, the centroid for each specific area must be recalculated, because it is not placed in the center of the region.

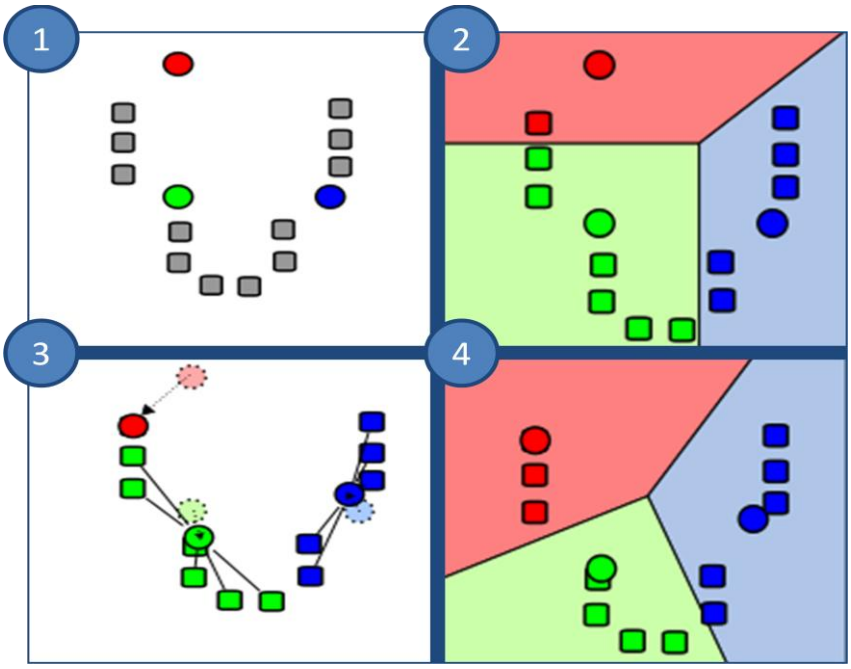


Figure 4: K-Means Process (Circles=Centroids, Squares=Codewords)

4. Iteration: Repeat steps 2 and 3 until the average distance does not reduce less than 0.001% from the previous distance.

The shape of each cell, sector or partition is singularly dependent on the spectral distortion and statistics of the vectors in the training group. This method is very simple and therefore it presents some weaknesses:

·The results depend sharply of the initial values chosen as codewords.

·There is highly dependency on the number of sectors M and the method to measure distance.

·It may happen that some of the sectors result empty.

LBG algorithm

It is a procedure which combines binary division with K-Means algorithm. It is a faster technique which guarantees no empty clusters and more equitable distribution of the clusters. It was developed by Linde, A. Buzo and R.M. Gray [5] and it is described as followed:

1. Initially include all the feature vectors in only one group (codeword).
2. Calculate the average/centroid of the codewords. (Calculate the error or average distance between the initial codewords and training vectors)
3. Doubling the size of the codebooks by splitting each codeword in two. K-Means algorithms should be used in this step to take the best group of centroids in the separation of the codebook.
4. Iterate steps 2 and 3 until a codebook of size M or with the required distortion. Observation vectors are grouped around the codewords with which they have shorter distance. Codewords are recalculated as the average of each sector multidimensional and data are grouped again.

This algorithm has been built following all the steps presented above. As result, it was obtained a full Codebook in a faster and more effective way than with the K-Means with apparently more equilibrated number of vector in each cluster algorithm. However at the end, the obtained classification was not accurate enough. In the testing phase, some similar feature vectors where generated when one speaker pronounced vocals in 5 different ways similar between each other.

The result was that the integers assigned for each pronunciation differed quite much from the others. This method was not valid enough for the isolated word recognition because it divides the space in the most equitable way but not in the most effective. Even if there are not enough different phonemes to complete the whole space, the method will split cluster which belong to the same phoneme in two or four until fill the space as it is specified.

3.2. Feature Matching

Once feature extraction has been completed and from an audio input of Nx32 seconds the system has gotten a Nx13 Matrix of features, it is necessary to face the recognition part. The question that summarizes the problem is clear: *How is possible to determine if the extracted features correspond to the pronunciation of a key word?*

The solution is not trivial at all. First of all, it is completely necessary to count with a set of references of each key word or command. The system should count with a library of different pronunciations for each command, where each command should be pronounced by different speakers, in a different context and with different tones in order to guarantee the best possible references to compare with the input audio. Better and larger library of sounds and pronunciations will increase the accuracy of the recognition.

The main problem is that the input and any library model cannot be directly compared due to the huge variability between vector components even if the vectors are similar and represent the same phoneme.

This variability is inherent to the characteristics of language. When a person speaks several parameters make each sound unique and unrepeatable: word duration, tone, timbre, rhythm, intensity, or environmental noise.

To overcome all this difficulties and to get accurate comparisons, the most useful and effective algorithms are presented:

- **Dynamic time warping (*DTW*)**
- **Hidden Markov Models (HMM)**

3.2.1. Dynamic time warping (DTW)

This algorithm was proposed in 1978 for Sakow, H and Chiba, S in a paper called Dynamic programming algorithm optimization for spoken word recognition [6]. Its main application is to measure similarity between two time series which may vary in time or speed. This algorithm is able to find the optimal alignment between two times series if time series may be warped non-linearly. The algorithm finds the optimal alignment by measuring and minimizing the distance between two time series. Figure 5 shows how one series is "warped" to another.

Each vertical line of the figure connects similar points between series. The second signal has been moved down to see clearly the alignment, but in reality the *y- coordinates* of connected point are very close. After warping together the two time series, the path distance is measured by adding the distance between each pair of connected points. This measure of the similarity between two dynamic patters by calculating the minimum distance between them is the principle of the DTW algorithm.

The algorithm computation is presented below.

Given two time series *Q and C, of length n and m respectively, where:*

$Q = q_1, q_2,\ldots, q_i,\ldots,q_n$ *(1).*

$C = c_1, c_2,\ldots, c_j,\ldots,c_m$ *(2).*

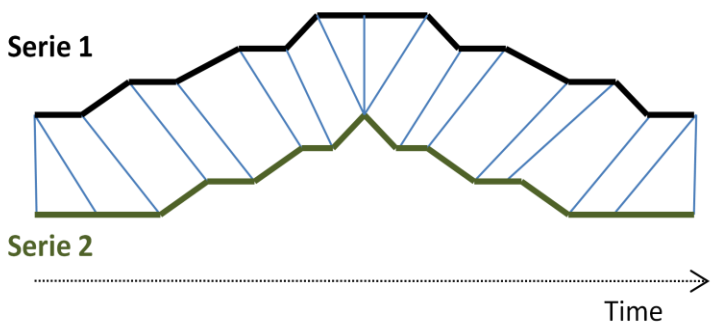


Figure 5: Time warping between signals

1. Build an n-by-m matrix where the (i_{th}, j_{th}) element of the matrix contains the Euclidian distance $d(q_i,c_j)$ between the two points q_i and c_j.

$$d(q_{i,cj}) = (q_i - c_j)^2 \quad i=1..n \quad j=1..m$$

2. Build the accumulated distance matrix D(i,j) in a dynamic process defined as:

$$D(i, j) = min[D(i-1, j-1),D(i-1, j),D(i, j-1)] + d(i, j)$$

$$With \;\; D(0,0)=d(0,0); \qquad D(1,0)=d(1,0); \qquad D(0,1)=d(0,1)$$

3. **D(n,m)** is the quantified *minimum distance* between vectors. It will give a measure of how similar two temporal series are. As lower this value is, as similar the series are.

It is possible to follow this process by means of a graphical representation: The soft lines mark the followed steps in the process. The darker line marks the optimal alignment or path that must be followed to get the minimum distance.

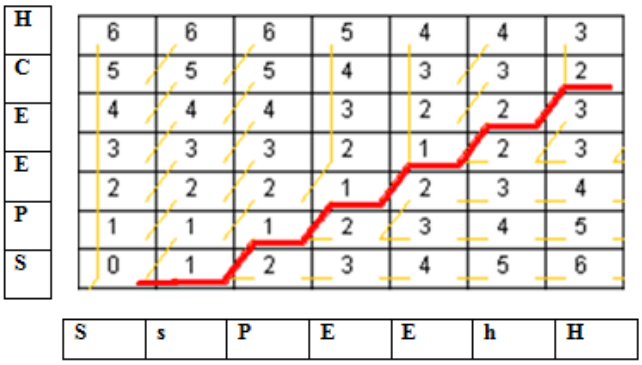


Figure 6: DTW quantifying process

In the specified graph, each sequence has to be arranged on different sizes of a grid. The unknown sequence will be placed in the bottom, while the reference or template to compare with, will be placed in the left part. Both sequences are considered to start on the left-down part of the grid. In each cell of the grid is written the distance measure between elements as this is calculated according to D(i,j) formula.

The optimal match between series will be given by the path which minimizes the total distance between series. This complex task of finding the optimal path between so many possible paths is shortened and simpler considering some optimizations of the DTW algorithm introduced by Sakoe and Chiba:

- *Monotonic condition*: the i and j indexes stay the same or increase. They can never decrease turning back in the path.
- *Continuity condition*: the path goes forward one step at a time; i and j can only increase by 1 on each step.
- *Boundary condition*: path end at top right and begins at bottom left.

- *Adjustment window condition*: any valid path is likely to remain close to the diagonal. The maximum distance that the path is allowed to travel from the diagonal is the window length *r*.
- *Slope constraint condition*: The path should not be too sheer or too shallow to avoid short sequences matching long sequences. The ratio n/m is the parameter that will determine this: n is the number of steps in the x direction and m is the number in the y direction.

DTW algorithm is an efficient tool since it keeps track of the cost of the best path to each point of the grid.

3.2.2. Hidden Markov Models

The Hidden Markov Model (HMM) is a recognized statistical model where the modeled system is considered to be a Markov process with hidden states. Algorithms and mathematics techniques that underline this model were developed by L.E Baum, T. Petrie, R.L Stratonovich and some other coworkers in the 60`s. It was not until the beginning of 1990 when Xuedong Huang introduces the HMM for Speech Recognition. [7]

Hidden Markov Models are widely applied in some other temporal patter recognition tasks, like in recognition of voice, music, gestures, faces, or writing. It has beneficial applications in bioinformatics areas.

In Speech recognition, an audio input is recorded and transformed into a matrix of features and then through the VQ algorithm this matrix is converted into a vector of integers which represents the sounds used in the speech. See figure 7.

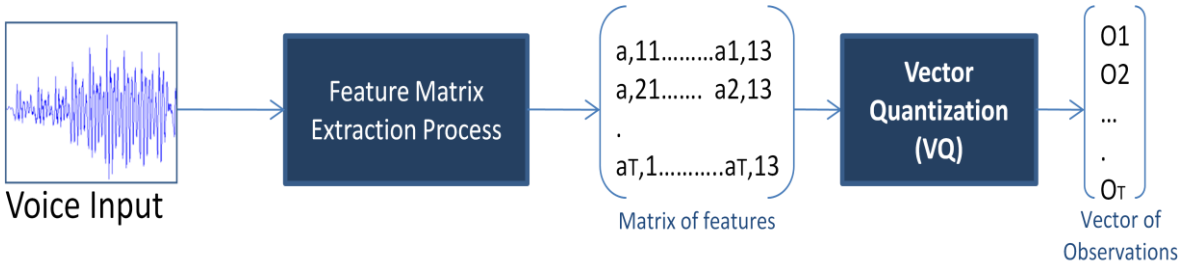


Figure 7: Extraction of Observation Vectors

HMMs allow estimating probabilities of unobserved events. For each key word, one HMM λi(A,B,π) is going to be built. Then the matching process will consist of calculating and checking which model has more probability to produce the given observations. Therefore, the observed data is the vector of observations O (obtained from the conversion of the audio input) and the states that generate them are the hidden parameters.

Given a sequence of observations O = {O_1, O_2… O_T} the last aim of the algorithm is to find the probability of each model λ to generate that sequence O $(P(O|\lambda))$. The model with the highest probability will be the best candidate to be the pronounced word. But before being able to compute this likelihood it is necessary to define and determine properly the parameters of the model.

The first concepts to define are:

- The number **N of states of each model**. Each model will represent a keyword, and since each keyword pronunciation is different, the number of states should be the optimal according to the length of each word.
- The number M of different possible observations that the model can generate, it is the discrete alphabet size.
- The type of HMM that it recommendable to be implemented. Two main possibilities:

  -Ergodic: Allows all the transitions from any state to any other state. Each state can be reached from any other state in a finite amount steps.

  -Left-Right HMM: Its peculiarity is that it always advances in time. The index of states is always moving forward in the time. It can remain in the same state, but it can never go backwards. It is a characteristic model for model signals whose properties vary with time, such as voice. Therefore, it will be the chosen architecture for the word models.

A HHM model $\lambda i(A,B,\pi)$ can be totally defined by three parameters:

-A, the state transitions probability matrix, Matrix $N \times N$ where

$$A(i,j) = P(\text{state } q_j \text{ at } t+1 \text{ state } q_i \text{ at } t)$$

-B, the observations probability matrix, Matrix $N \times M$ where

$$B(j,k) = b_j(k) = P(\text{observation } k \text{ at } t \; j \text{ state } q_j \text{ at } t)$$

-π, the initial state distribution vector, Vector of N components, where

$$\Pi(i) = P(\text{state } q_j \text{ at } t=0) \qquad If\ i=1 \quad \Pi(1)=1, \ else\ \Pi(i)=0$$

Since the chosen model is the left-right HMM, the process will start always on the left moving forward to the right.
A graphical representation of an HMM model of six states, left-to right model is shown below:

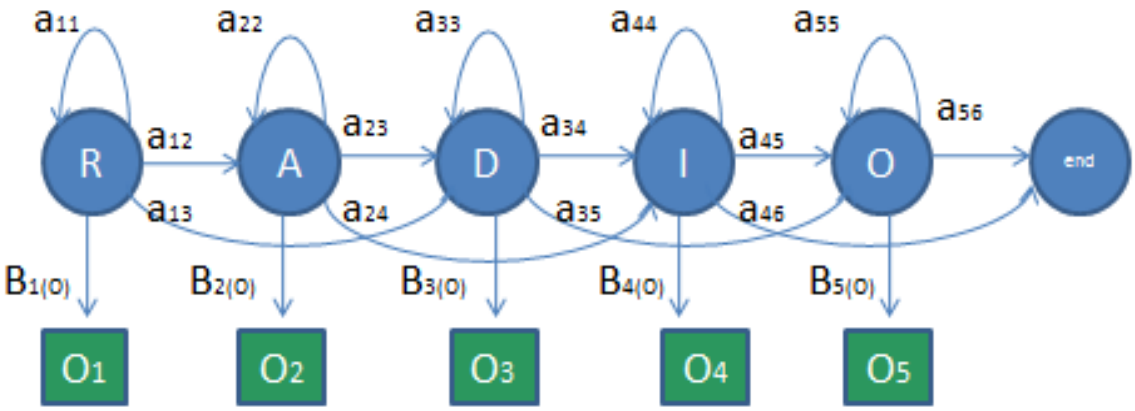


Figure 8: HMM left-right model with 6 states

HMM three basic problems

Three basic problems must be solved for the model in order to be useful in real applications:

1. Given the sequence $O = \{O_1, O_2 \ldots O_T\}$ and the model $\lambda(A,B,\pi)$, find the probability of this model to generate that sequence $(P(O \mid \lambda))$

2. Adjust the parameters of the model (A, B, π) to have the most representative and accurate model. It is called the Training problem.

3. Given the sequence $O = \{O_1, O_2 \ldots O_T\}$ and the model $\lambda(A,B,\pi)$, find the optimal sequence of states $q_1 q_2 q_3 \ldots q_T$ which produces the observation sequence O with maximum probability.

Solutions to the HMM basic problems. HMM algorithms

Problem 1

The problem is Giving the sequence $O = \{O_1, O_2 \ldots O_T\}$ and the model $\lambda(A,B,\pi)$, find the probability of this model to generate that sequence $(P(O \mid \lambda))$.

The way simplest without applying any particular procedure is through the numbering of each possible sequence of states of length T (number of observations). Consider the following mixed sequences: $Q = q_1 q_2 \ldots q_T$. Then the probability of the observation sequence given the previous state sequence is the following equation:

$$P(O \mid Q, \lambda) = \prod_{t=1}^{T} P(O_t \mid q_t, \lambda)$$

Where it is assumed the statistical independence of observations, so:

$$P(O \mid q, \lambda) = b_{q1}(O_1) \cdot b_{q2}(O_2) \ldots b_{qT}(O_T)$$

The probability of each static sequence Q can be written as the following expression:

$$P(Q \mid \lambda) = \pi_{q_1} * a_{q_1 q_2} * a_{q_2 q_3} ... * a_{q_{T-1} q_T}$$

And taking into account that

$$P\ (O,\ Q \mid \lambda)\ = P\ (O\ \mid Q, \lambda)\ P(Q \mid \lambda)$$

Then:

$$P\ (O,\ Q \mid\ \lambda)\ = \Sigma\ P\left(O \mid Q,\ \lambda\right) * P\left(Q \mid \lambda\right) = \sum_{s=s_1, s_2 .. s_T} \pi_{s_1} B_{o_1, s_1} A_{s_2, s_1} ... A_{s_{T-1}, s_T} B_{o_T, s_T}$$

The interpretation is as follows: initially (at time t=1) the system is in the state q_1 with probability equal to $\pi \times q_1$, and generates the symbols O_1 (in this state) with probability $b_{q1}\ (O_1)$. When the time changes from t to t +1 (time t=2) the model performs a state transition from state q_1 q_2 with probability a_{q1q2}, which generates the symbol with probability $b_{q2}O_2$. This process continues like this until the last transition from state q_{T-1} *to* q_T with a probability of $a_{qT\text{-}1qT}$, which generates O_T with a probability of $b_{qT}(O_T)$.

A short analysis verifies that the calculation of $P(O|\lambda)$ involves the order of $2T \times NT$ calculations. Clearly a more efficient procedure is required to solve this problem, and fortunately such procedure exists and is called the forward procedure.

Forward - backward procedure

The named technique takes into account that some sub paths are common to many continuing paths, and it uses dynamic programming to calculate and add partial probabilities until the total probability is computed.

The algorithm, therefore, iteratively re-estimates the parameters and improves the probability for the model with new parameters to generate the given observation.
With this procedure, it is possible to find a local maximum, but it depends on the initial estimation.

-The forward variable α(i) is definite like the probability of being in state s_i, given the partial sequence of observations $o_1, \ldots, o_T$.

$$\alpha_t(i) = p(o_1, o_2, ..o_t, s_t = S_i \mid \lambda)$$

In Zhang, 1999 it is described how each value of the vector is calculated iteratively:

i) Initialization $\alpha_1(i) = \pi_i B_{o_1 . i}$ $1 \le i \le N$

ii) Induction $\alpha_{t+1}(i) = \left[\sum_{j=1}^{N} \alpha_t(j) A_{ji} \right] B_{o_{t+1}, i}$

Finally, the probability of this model to generate that sequence will be given as:

iii) Termination $P(O \mid Q, \lambda) = \sum_{i=1}^{N} \alpha_T(i)$

-The backward variable β(i) is definite like the probability of being in state s_i, given the partial sequence $o_{t+1},\ldots,o_T$

$$\beta(i) = p(o_{t+1}, o_{t+2}, ..o_T, s_t = S_i \mid \lambda)$$

The components of this vector are also calculated in an iteratively way:

i) Initialization $\beta(i) = 1 \qquad 1 \le i \le N$

ii) Induction $\beta_t(i) = \sum_{j=1}^{N} A_{ij} B_{o_{t+1}, j} \beta_{t+1}(j)$

iii) Termination $P(O \mid Q, \lambda) = \sum_{i=1}^{N} \pi_i B_{o_1, i} \beta_1(i)$

Problem 2

To get the most accurate model for each key word, it is necessary to choose carefully and train with a precise method the parameters of the model. Maximization Estimation algorithm will be taken as a reference to train models since this procedure tries to maximize the probability for the model to produce the observation sequence.

It is essentially an iterative process that locally maximizes the probability that an observation sequence was generated by a particular model [P (O | λ)] and in a way that ensures the convergence of the process from an initial random model.
One of the most popular methods to accomplish this task is the process of maximizing estimation EM, and as a specialization of the same, the algorithm of Baum – Welch, described below:

1. **Obtain the initial model:** The initial Matrix A and Matrix B are obtained in a semi random way, but always presumably subject to the restrictions of common odds. There is the possibility of implementing improvements that will help obtain more accurate initial parameters, but they were avoided since the results with the considered initial matrix are good enough. Anyway, it should be noted that as closer to the global maximum the initial model is, as more accurate the final model will be.

2. **Calculate the probability** that the observation has been generated by the obtained model. It is done by the auxiliary intermediate prior algorithms that reduce the computational complexity of the process (forward and backward algorithms). This probability is calculated for each of the models obtained to verify that it is maximal.

3. **Re-estimate the model:** recalculating the model parameters using the above formulas, relying on the model obtained in the previous iteration.

   It is necessary to define the expected probability of the transition from state S_i to the state S_j at time t with the sequence o. It is computed making used of the forward and backward variables:

$$\varepsilon_t(i,t) = p(s_t = S_i, s_{t+1} = S_j \mid \bar{o}, \lambda) = \frac{\alpha_t(i) A_{ij} B_{o_{t+1},j} \beta_{t+1}(j)}{\sum_m \sum_n \alpha_t(m) A_{mn} B_{o_{t+1},n} \beta_{t+1}(n)}$$

   Then the expected likelihood of transition from i at time t is:

$$\gamma(i) = \sum_{j=1}^{N} \varepsilon_t(i,t)$$

   Hence, the equations to calculate the new model parameters are:

$$\overline{\pi_i} = \textit{expected visits number in state i at time 1} = \gamma_1(i)$$

$$\overline{A_{ij}} = \frac{\sum_t \varepsilon_t(i,j)}{\sum_t \gamma_t(i)} = \frac{\textit{expected number of transitios from state i to j}}{\textit{expected number of transitios from state i}}$$

$$\overline{B_i(k)} = \frac{\sum_{t:o_t = v_k} \gamma_t(i)}{\sum_t \gamma(i)} = \frac{\textit{expected number of emissions of observation } v_k \textit{ from statet i}}{\textit{expected number of transitios from state i}}$$

4. **Probability re-estimation and optimal model:** After getting new parameters for the HMM, it is necessary to calculate the probability that the observation was generated by the new model. If this probability has reached a maximum (what will be checked in the next iteration) or if the number of iterations is big enough to have an accurate model, then the model is completed trained and saved for its posterior use.

Problem 3

The problem of finding the optimal sequence of states $q_1 q_2 q_3 \ldots q_T$ which produces the observation sequence O with the maximum probability, in principle is not concerning for the voice recognition task. What it is important to know is which model generates the sequence of observation with the highest probability. A priori it is not relevant to know which optimal sequence of states generates the model. But since it is a useful mode to check if the model is correctly trained with a reasonable sequence of states according to the obtained observations, a brief summary of Viterbi Algorithm operation is presented.

i) Initiation:

$$\delta_1(j) = \pi_i b_j(O_1) \qquad 1 \le i \le N$$

ii) Induction:

$$\delta_{t+1}(j) = b_j(O_{t+1})\left[\max_{1 \le i \le N} \delta_t(i) a_{ij}\right] \qquad 1 \le t \le T-1$$

iii) Termination

$$p^* = \max[\delta_T(i)]$$

$$q^* = \arg\max[\delta_T(i)]$$

Chapter 4

Results & Practical Approach

This section first presents the general architecture and design process of the own voice control system. It then delves into the implementation modes of the speech recognition system, explaining in details characteristics and peculiarities of each implementation together with possible modifications of them. Finally, the document contains information and data relative to effectiveness, consumption, and accuracy of the system working in different modes and modifying diverse variables.

4.1. Design and architecture of the voice control system

The main objective of this project is to design, implement and test a basic voice-controlled low-power radio for sports communication.

Due to the high cost in knowledge, time and money of designing and manufacturing an own VHF radio with voice control system, it was raised the acquisition of commercial a VHF radio that could be controlled from an external portable host terminal. This portable terminal could be manufactured what supposed too much work for only one student or it could be purchase after studying different alternatives of the market. One nice and comfortable solution was to use a Smartphone as this host terminal since new devices usually allow to program attractive and useful applications to manage its resources. Chalmers Computer Science Department was asked about this possibility and they offered as a possible host terminal a Samsung Galaxy S Plus. The characteristics of this model were analyzed and it was finally

accepted since it is a 2.3.3 Android terminal open to Java programming through Android SKD, what implies an easy manage of almost all the resources of the phone from intuitive applications. Note that studied documentation also introduced the possibility of re-programming the kernel of the phone to extend the range of possibilities of the system. (USB host mode).

While developing programming for algorithms of voice control in Android SDK was carried out, different market control remote VHF radios were studied in depth, checking amateur and professional devices which could meet the desired specifications. However, this finding was not easy at all; it was found a big lack of portable VHF radio with easy PC-control capability from an USB/serial or wireless port. The initial devices that were found presented several incompatibilities; they were not open programmed, they were not compatible with USB/serial or wireless communication, or they did not allow Push-to-Talk feature in control remote mode. After an exhaustive but fruitless search it was consulted to the company which model they were interested in work with, receiving like a desirable device the ICOM ID-31 E. See features and specifications in reference [8]. This model was requested to the Chalmers University of Technology and finally it was purchased for this aim.

The ICOM ID-31E radio is a handled amateur UHF digital transceiver from ICOM which not only allows remote radio control, it also presents a long list of features which place this model at the top of the market. Icom has not any similar VHF model, so an UHF device which works between 430-440 MHz, with an output power of 5W was accepted as recommendable taking into account that the power of the terminal cannot be modified and the main purpose of the thesis is not related with the operative band of frequencies. The radio includes the CI-V serial communication interface to make the remote control possible.

After decided which radio and which terminal would work in the system, it was necessary to determine how the system could work. The first proposed approach was clear: try to control directly the radio from the phone implementing serial communication through the CI-V system. To archive it, the mobile phone needed to have host control capability, matter that initially the phone does not implement as standard.

Documentation process would lead us to know that reprogramming the kernel of the phone to allow USB host mode was not possible in the available Samsung Galaxy S Plus model. Besides getting or programming the driver for the phone to control the radio would be a taught task out of any requirement. Therefore, and knowing that it was not a very portable system, but considering that it was the only feasible solution for this demonstrator, a laptop was introduced in the system in order to get the host mode capability. The final architecture of the system is presented in the illustration 9.

Because of the impediment of manage USB communication from the Smartphone, two communication system were developed in the system trying to get the maximum portability. Maximize learning experience in communication protocols was also important.

The TCP/IP provides reliable, ordered delivery of a stream of bytes from the android program to the PC program. By using TCP/IP instead of UDP it is possible to guarantee data arrives in the correct order with safety what is decisive to make the correct command arrives to the radio.

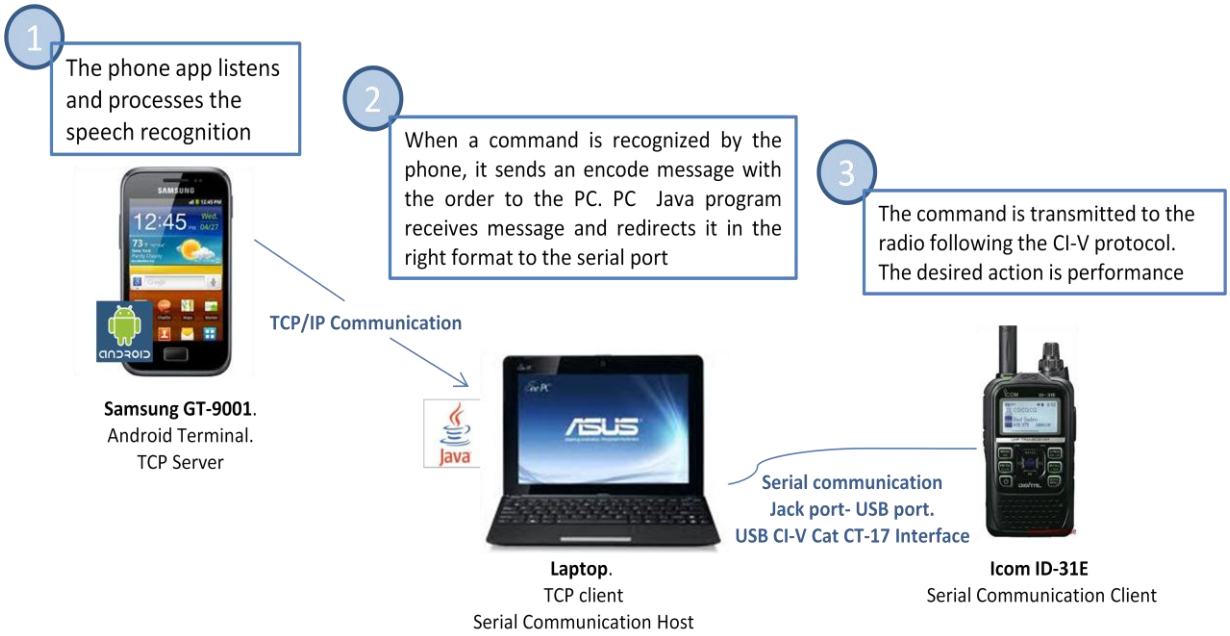


Figure 9: System architecture

The Icom radio just accepts serial communication through CI-V communication interface with CSMA/CD System for avoiding collisions. The communication is established between PC and radio by the specific cable USB Cat CT-17 which makes an USB port act as a RS-232 serial port working at the same time as a level converter. This communication often presents difficulties due to manufacturing faults in radio jack port. See reference [8] to get informed about how CI-V is operated.

The selected configuration for the system is a compromise solution between functionality and ease of implementation. Although portability of the system is not adapted to sportive uses, the system has a comfortable user interface and it is easily adaptable to other models of Android Mobile phones and Icom Control Remote Radios. The performance of the whole system is good, presenting high rates of accuracy, speed and robustness.

4.2. Speech recognition system: Design and Implementation

The speech recognition process is completely run by the programmed Android Application. It has been designed to work totally independent of any other component of the whole voice control system constituting by itself a completed speech recognition system which can be easily adaptable for other applications in other platforms. The ASR is able to listen and identify adaptable specific commands within almost any kind of conversation.

The study, design development and testing of the command recognition system has taken around five months of intensive work and as a result two different approaches for voice recognition are presented: **DTW algorithm and Hidden Markov Models.**

These algorithms differ quite much in concept and operation but signal extraction process and their aim are the same: rate how similar two sounds are. Therefore, and although there are two different algorithms and subsequently two different Android Applications, the

general structure of these programs is the same; they follow the same flowchart with variations only in the matching process. See flowchart in figure 10.

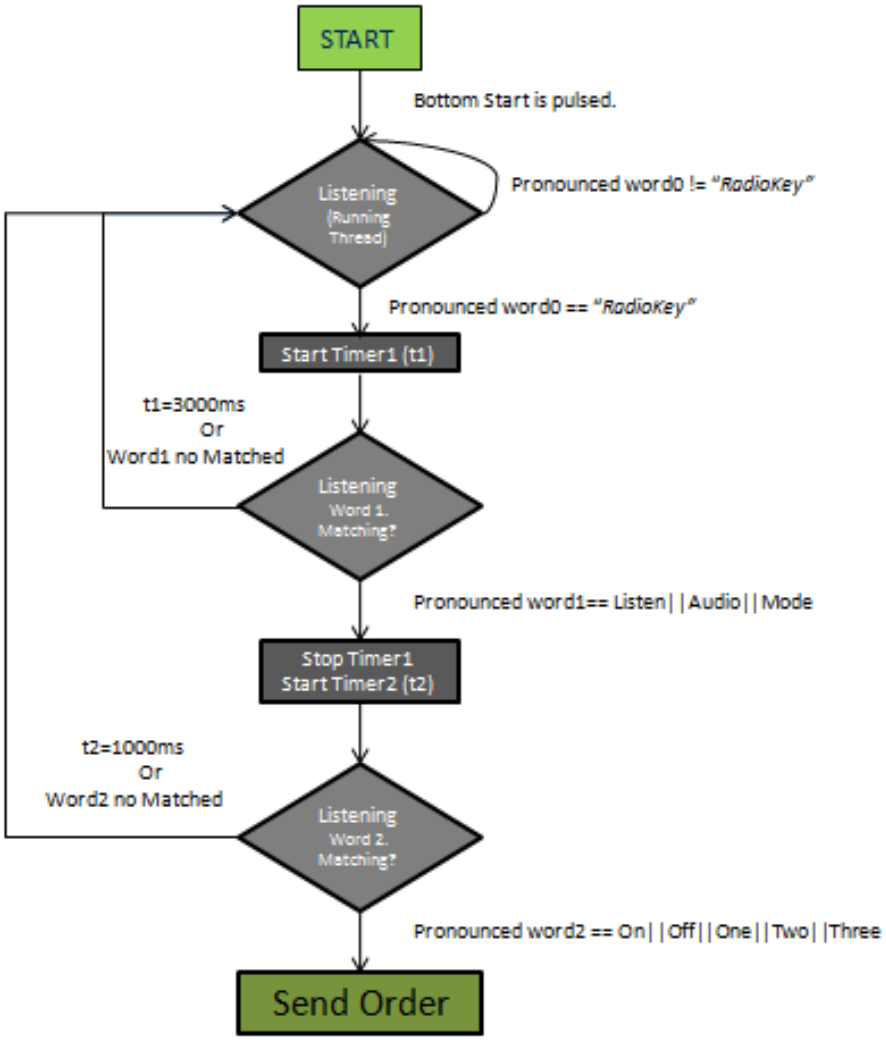


Figure 10: Program Flowchart

Either VoiceCrtl_HMM or VoiceCrtl_DTW works only with two bottoms (start and stop) and prefixed voice commands are composed by two words. The main idea for both systems is before saying the desired command to the terminal it is necessary to pronounce a key word like “Radiokey”. With this method, probabilities of success increase since just one special word has to be differentiated within a completed set of possible sounds in a noisy environment. If the system recognizes this key word, then it will wait for three seconds until the word 1 of the command is pronounced. If the pronounced word matches well enough with one of the three possible words1 (mode, listen, audio), then the system will proceed in the same way waiting for the word 2 of the command for one second more. (Possible words2: on, off, one, two, three). At the end, if the matching of the two words is good enough, the recognized command will send it to the radio.

By combining these two different words to form the global command, the system performs the recognition of two words instead of one, what gives to the system higher accuracy avoiding misunderstanding and confusion.
In the following table is presented what each command means in the Icom Radio instructions language:

Tabla 1: Radio Commands

Word1 / Word2	On	Off	One	Two	Three
Listen	Vox On	Vox Off	Sensibility 1	Sensibility 2	Sensibility 3
Audio	Squelch On	Squelch Off	Audio Level1	Audio Level2	Audio Level3
Mode	Power on	Power off	FM Mode	FM-N Mode	DV Mode

4.2.1. Dynamic Time Warping Implementation

Keeping in mind how command recognition system works and reviewing the basis on Automatic Speech Recognition presented in the section 3.1, it is easy to see that the main problem of command recognition is the comparison between the current listened sound and the reference sounds saved in libraries. It is worth remembering that each listened sound that has enough energy for more than 0.2 seconds constitute a potential command that first is recorded and then is processed to obtain its vector of features. Then this vector of features will be matched by the Dynamic Time Warping Technique with all the features vector saved in the library in order to see how similar the characteristics of the recorded sounds are from each command characteristics. There have been developed two methods to perform this matching:

·**Individual Matching:** The pronounced command is compared one by one with all the feature vectors of the library. The recognized word will be associated to the feature matrix with a high rate of matching. By this procedure, it really does not matter if the matching with another repetition of the same word is very low. The highest matching corresponds only to a specific vector, no matter how good the matching with other repetitions is.

·**Average Matching:** The pronounced command is matched by making an average of individual matchings between all the repetitions of the same word. This procedure consists of matching the pronounced word with all the repetitions of a keyword and then obtain the average score matching value for that key word. At the end, the group with high average will be recognized as the pronounced word.
In accordance with the previously introduced idea, the initial step to implement the DTW Technique is to create a library of vectors of features, where each vector represents the features of a word pronunciation reference. Or what is to say, initially it is a requisite for the final user to record each word command pronunciation at least once in order to have a reference to match

with. Then these recordings will be transformed in the corresponding vectors of features that will compose the library by using the Vector Quantization Technique.

For the VoiceCrtl_DTW program several tests have been performed trying to see how different combination of system variables affects to the accuracy and response time of the system. There are three main fields where variations can be introduced:

·**Number of MFCC** coefficients with which the program works.

·**Number of repetitions** for each command word (the number of feature vectors associated with a command word).

·**Matching mode:** Individual or average matching.

Playing with these variables, tables in page 38 reflect measurements and data of rate error and time response time or the system.

Note: The response time is the time that the system takes since the potential command is recognized until the action takes place in the radio. It is measure manually with a digital chronometer thus it is just a pure estimation of the speed system. The rate of error of the system could be defined as the percentage of error that the system fails when a correct command pronunciation is performed. It could be rated in many different ways, but it was decided to design two possible simple and intuitive formulations that easily allows getting an idea of the accuracy of the system.

The first approach considers that to transmit successfully an order to the radio three words must be correctly recognized ("RadioKey+WordCommand1+WordCommand2) and that there are 15 possible commands one possible formula could be:

$$Error\ (\%) = (1 - (0.25 * \sum_{N=1}^{4} \sum_{i=1}^{i=15} \frac{Number\ of\ reconginited\ words\ in\ command\ i}{3 * 15})) * 100$$

The other formula would only assess the correct global commands that the system recognizes, therefore at first this is the method chosen to measure the global system error. Results will be rounded to the nearest unit.

$$Error\ (\%) = \left(1 - \left(0.25 * \sum_{N=1}^{4} \sum_{i=1}^{i=15} \frac{R}{15}\right)\right) * 100$$

Where R=1 if the total command has been correctly recognized.

Table 2: Accuracy of the DTW algorithm as a function of the number of MFCC. The accuracy test was developed **for the trainer of the system.**

	Matching Mode		13 MFCCs		39 MFCCs	
			Time(ms)	Error %	Time(ms)	Error %
Number of Repetitions	5	Average	903	37%	2644	78%
		Individual	**946**	**28%**	2577	64%
	15	Average	2974	27%	6864	59%
		Individual	2816	28%	6763	46%
	25	Average	4120	23%	11624	56%
		Individual	3980	30%	11577	53%

Table 3: Accuracy of the DTW algorithm as a function of the number of MFCC. The accuracy test was developed **for trainers outside of the system.**

	Matching Mode		13 MFCCs		39 MFCCs	
			Time(ms)	Error %	Time(ms)	Error %
Number of Repetitions	5	Average	903	98%	2644	100%
		Individual	946	97%	2577	100%
	15	Average	2974	91%	6864	100%
		Individual	2816	91%	6763	98%
	25	Average	4120	89%	11624	100%
		Individual	3980	95%	11577	100%

In view of the presented results it is easy to check library commands should be pronounced by the final client since DTW technique does not create a model for each word, it just compare audio features. If the voice used to record the library is not the same than the client, then features will further differ between them and as a consequence the efficiency of the system will be reduced to almost zero.

Note that the matching process between two feature matrixes is described in 3.3.1, so in this section just information and data related to performance testing has been included.

Analysis of the results

As it is normal and clearly visible in the results, processing time increases with the number of matching that the terminal has to process. Time increases linearly with the number of repetitions that the system has to match. The expended time matching 5 repetitions per word is around one second, fact that is not optimum but assumable for comfortable user experience. However, higher processing time than 1.5 or 2 seconds implies confusions and waiting time too long to have a handy system interaction.

Note also the difference in time between the use of average technique or individual matching, it is irrelevant, so for the accepted case with 5 repetitions, the individual technique will be chosen since it presents higher accuracy, around 72%.

The dissimilarities between average and individual technique are notable; accuracy with the average technique always increases with the number of repetitions. It could be explained since the program is calculating the average matching between the input and all repetitions of a word. As many repetitions the model has, as less variability the global matching will present, and therefore a more accuracy system will be obtained.

On the contrary, with the in the individual technique number of repetitions influences accuracy differently: when few repetitions are used, the exactitude of the system is even higher than with the average technique if the input coincides highly with a repetition. However, when too many repetitions are used, the input can be falsely matched with a no very precise repetition of another word. That is because if a more precise system needs to be implemented without considering the processing time, then a system with 25 repetitions with average matching will be chosen.

The number of coefficients which compose the feature vectors after audio processing (MFCC) is very relevant. Contrary to what might be intuitive, the best results are obtained working with fewer feature coefficients. Increasing the number of coefficients until 39, by adding coefficients related to the first and second derivate of the initial 13 has a significant negative effect on the system accuracy. This can be explained from the fact that 39 coefficients define well the audio input characteristics and its dynamic when the words have more than two syllables. However, the characteristic features of monosyllable words as *on, off, two*, are not clearly extracted with 39 features method, since in monosyllable case dynamic characteristic are less appreciable and the method estimate wrong coefficients to explain them. What is more, working with so many coefficients makes the recognition process too slow and inefficient, with processing time until 10 seconds. The use of vectors with 13 MFCC is so far fully justified.

Optimal results are gotten for one speaker use, when the matching mode is performed among 5 repetitions per word. System accuracy is around 75%, which is considerably high for such elemental procedure.

4.2.2. Hidden Markov Implementation

Unlike the DTW matching technique, use of Hidden Markov Model will allow multiple speaker commands recognition. This technique is based on the creation of particularized state models for each word accepted by the system. Once the system has at least one model for word, matching process consist of checking which model has the highest probability of generating the observation vector associated with the input audio.

This section first introduces particularities and explanations of how models are created. Then results and statistics of different type of models performance are shown. In a more advanced stage, the question of obtaining the observation vector is analyzed, showing characteristics and results of different approaches. In the core of the section, the training process of each model is deeply studied.. Due to the relevance of this procedure in accuracy and processing time the study has been divided in two phases trying to follow some guidelines that were improving system performance step by step.

In a first stage, the adopted method consists in the study and improvement of the recognition of commands from a single speaker who had previously participated in training the models. This phase includes, among others, studies about ideal number of states per model, number of Codebook clusters, or multi-sequence parameters.

Once the basic training parameters were known from the individual training, the second phase the study included the training of the models by different speakers with the aim of increasing the number users recognized by the system, guiding the training to universal recognition. Several graphs, tables and statistics show the progress and effectiveness of the system with each implemented step.

Appreciations and preliminary conclusions are detailed to conclude the chapter.

a) Mono-Trainer. Training Models by one speaker

a.1) Studies in Word Model Design: Number of states in each model

As it was already explained in chapter 3.3.2, each HMM is considered as a statistic model built from hidden states, which can be completely defined by three parameters $\lambda=f(A,B,\pi)$.

Regarding the two main possible architectures for the models, the Left-Right structure will be always the working constant architecture for the chosen models, since the transitions between states are always forward as the theory describes.

As a basis for correctly defining each model, once the structure is appropriately selected, it is essential to choose optimally the number of states that each word be associated model. The optimal number N of chosen states will depend on the model parameters (A,B, π) and therefore it will affect the effectiveness and efficiency considerably.

In an initial configuration, it was decided to work with the same number of states for all the words to see how the system responded. It was required to study if specific adaptation to the word length was essential and how an incorrect number of states influenced on the accuracy of the system. Trying to find a compromise solution for all words, 6 states models were developed. One user' voice has been used for training the models and the same voice has tested the system.

To measure the accuracy, each word was tried to be recognized 25 times. The corresponding codebook was trained to get 64 clusters with 20 repetitions of each word for each speaker.

Table 4: Accuracy of the HMM system with six states per model.

	Command Words								
	RadioKey	On	Off	One	Two	Three	Listen	Mode	Audio
Nº of States	6	6	6	6	6	6	6	6	6
Accuracy (%)	97	96	85	100	98	100	100	100	100

As the data shows, the fidelity of this procedure is not quite high, but it still forces the research to try to increase the accuracy of the system by adapting the number of states of each model to the number of sounds and length of each word. After more than 20 trials and adaptations, the optimum configuration was found with the following distribution of states. Accuracy, code extension and processing time and use of memory are measure. Vector quantization has been developed by K-Means algorithm with 64 centroids. Results are published in table 5.

Table 5: Accuracy of the HMM system as a function of the number states per model.

	Command Words								
	RadioKey	On	Off	One	Two	Three	Listen	Mode	Audio
Nº of States	6	3	3	3	3	3	4	4	3
Accuracy (%)	97	100	98	100	100	100	100	100	100

As it is observable, improvement in system accuracy is remarkable in the models of a single syllable. This phenomenon can be justified from the fact that, in some short pronunciations of this mono-syllable word, there are not enough observations to complete the corresponding HMM and it is there when the recognition fails.

In general, the results keeping the 6 states for all the words are quite acceptable. Regardless, adaptive model will be use from now on to ensure better accuracy and reduce processing time.

a.2) Studies in Vector Quantization Technique: Number of clusters and iterations.

Found that adapting the number of states to the number of phonemes in each word, increases global system efficiency by up to 8%, now it is possible to research about the influence of Vector Quantization variables in the speech recognition process. Each HMM has as known parameters the observation vector obtained by the conversion of a matrix of features into an observation vector by the Vector Quantization technique. The transformation of each vector

from the matrix into an integer number can be compared as the process of allocation each number (which defines a phoneme) to each interval of 0.032 seconds. The total range of numbers (or what is comparable, the total number of phonemes that the system distinguishes) matches the total number of clusters of the design Codebook. To convert from audio to observations, several types of Codebooks have been developed and trained base on the VQ theory. See clarifier figure11.

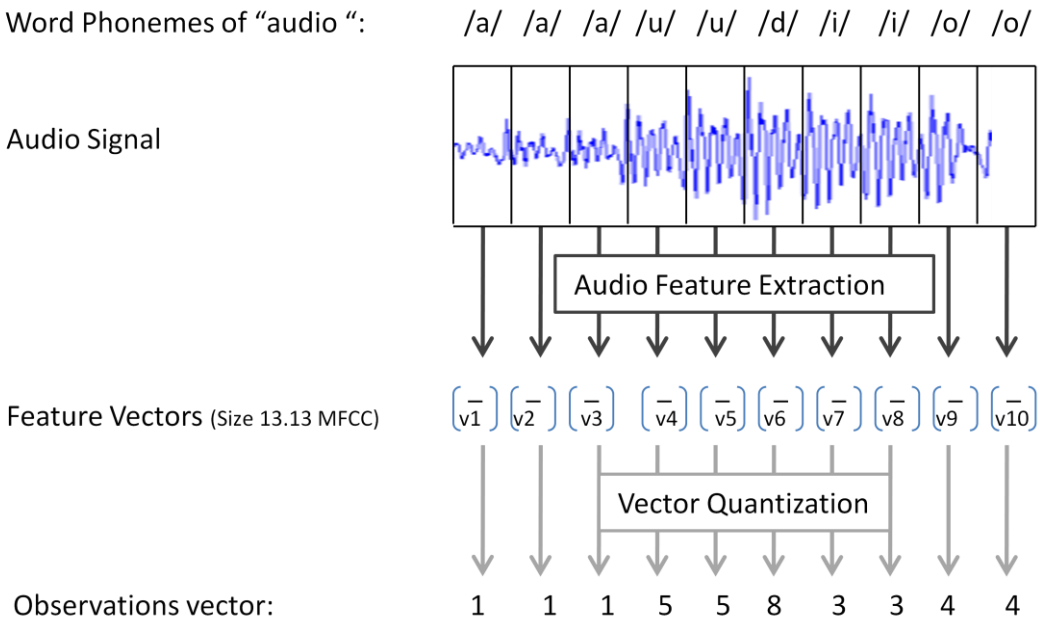


Figure 11: Process from the audio signal to Observation Vector

Regarding classification algorithms for training the Codebook, K-Means and LBG algorithms were implemented. However, it must be reported that although LBG processing is around four or five times faster on average than K-Means, the obtained results with LBG were not expected, resulting unsatisfactory for the recognition use. The LBG algorithm was reviewed several times, without finding any relevant incorrectness. Further researches need to be done on this, but it may be possible to adventurer the main reason for this phenomenon: the algorithm divides the space in the most equitable way, even if there are not enough different phonemes to complete the whole space, it will split clusters which belong to the same phoneme in two or more parts until the space is filled as the algorithm specifies.

The table 6 in the next page shows the accuracy of the system working with observation vectors of 16, 32, and 64 vector components or Clusters (≈ Distinguished phonemes). It also reflects the needed time for the deviation to converge into a specific value. Experiments were developed only with one voice recording repetitions (15 or 30 reps). The first approach in multi-sequence training was effective. Accuracy is measure using:

· Mono-voice: The same voice that trained is used to estimate the accuracy of the recognition. The accuracy achieved in the best configuration is acceptable, over 94%.

Table 6: Recognition accuracy, processing time and specific Codebook parameters when the codebook get composed by 16, 32 or 64 clusters

Number of Clusters	Codebook Data		Codebook Training			Recog. Accuracy		Recog. Processing
	Repetitions/word	Total N° samples	Deviation Convergence			Same Speaker (Error %)	Any Speaker (Error %)	Processing Time (sec)
			Value	Minutes	Iterations			
16	15		211700	47	22	45%	95%	0.3
	30		384137	185*	19	15%	80%	0.3
32	15		193460	34	16	40%	90%	0.42
	30		343955	112	20	10%	75%	0.42
64	15		172355	56	20	40%	90%	0.67
	30		313038	135	18	3%	70%	0.67

Table 7: Recognition accuracy, processing time and specific Codebook parameters when the training is performance by one, two or three different speakers

Number of Speakers	Codebook Training				Recog. Accuracy		Recog. Processing
	Total N° samples	Deviation Convergence			Same Speaker (Error %)	Any Speaker (Error %)	Processing Time (sec)
		Value	Minutes	Iterations			
1	14668	313038	135	18	3%	70%	0.67*
3	33590	626217	885	19	5%	30%	0.67
4	41082	896814	1732	22	60%	75%	0.67

· Multi-voice: New and strange voices for the system are use to measure the accuracy of the system. Since only one voice was used to train the model, the accuracy of the system is poorer, around 30% in the best case.

By this experiment, it is clear that the best configuration is reached with the maximum number of repetitions per word and the maximum clusters (30 reps and 64 Clusters).
Centroids with 128 Clusters could have been developed, but this would have meant an excessive processing time and it could then undermine the success of the HMM method. Results for multiple Codebook trainers are indicated in the table 7.

a.3) Studies in HMM training: Multiple Observation Sequences

Completed studies, about the ideal number of states for each word and about the optimal length of the observation vectors, have been developed in previous sections. The drawn conclusions and their application in the system allow a priori to ensure satisfactory results if the training of the models is correctly developed. Regardless the importance of this training is not trivial; it is the key problem in HMM implementation, and it determines the grade of success or failure of the system.

The objective of training is to obtain a final model $\lambda(A,B,\pi)$ for each command which maximizes the probability that observation training sequences were generated by this particular model [$P(O|\lambda)$]. Each particular model represents a word, and therefore each training sequences (O) must be nothing more than the training vector obtained from the processing of each word repetition. Since each model is desirable to be trained according to several repetitions, the main incognita now is how to train a single representative model that fits so many different reps.

The study of various sources [9], [10] has recommended the implementation and outcome study of two different approaches based on formulas that reaches a consensus between individual all the single-sequence models. Both procedures follow the same principle: to get an accurate multiple sequence trained model, it is necessary first to create and train a separate model for each repetition of the word by following single-sequence Baum-Welch techniques (See solution of the individual training problem in 3.3.2). Once these models have been created and optimized for each rep, a general model, which joins all the particular models, is obtained by applying each particular formula.

Before presenting these communion formulas, particularities of the individual training process are following described. (Note that the whole is explained in Rabiner [10] and summarized in the theory section 3.3.2). It must be reported that although initial matrixes are suppose to be obtained with a totally random distribution, several successful probes have been done with predetermined matrix, avoiding complex statistics and extra programming. Results show a high level of accuracy when the distributions above for the initial model are used.

Matrix Ai (N×N)

$$\begin{pmatrix} 0.3 & 0.4 & 0.3 & 0 & 0\ldots\ldots 0 \\ 0 & 0.3 & 0.4 & 0.3 & 0\ldots\ldots 0 \\ . & . & . & & . \quad . \\ . & . & . & & \\ 0 & 0 & 0 & 0 & .0.8 \quad 0.2 \\ 0 & 0 & 0 & 0 & 0 \quad \ldots 1 \end{pmatrix}$$

Matrix Bi (N×M)

$$\frac{1}{Ob} * I$$

Vector πi (N)

$$\begin{pmatrix} 1 & 0 & 0 & .. & 0 \end{pmatrix}$$

The training process for each particular repetition continues by applying the Baum-Welch algorithm with the Forward - backward procedures. In this manner, for each particular model several trained parameters will be calculated:

· Matrix A with a*ij* elements

· Matrix B with *bij* elements

· The forward variable α. α(i) elements

· The backward variable β. β(j) elements.

Now, assuming that K repetitions of a word are necessary to train the model, there are two main ways of estimating the general matrixes components:

1) $$\bar{a}_{ij} = \frac{\sum_k W_k \sum_{t=1}^{T_k} \alpha_i^k a_{ij} b_j(O_{t+1}^{(k)}) \beta_{t+1}^{(k)}(j)}{\sum_{k=1}^{K} W_k \sum_{t=1}^{T_k} \alpha_t^k \beta_t^{(k)}(j)}$$

$$\bar{b}_{ij} = \frac{\sum_k W_k \sum_{O_t(k)=v_j}^{T_k} \alpha_t^k(i) \beta_t^{(k)}(j)}{\sum_{k=1}^{K} W_k \sum_{t=1}^{T_k} \alpha_t^k \beta_t^{(k)}(j)}$$

W_k represents the weight of each individual model in the general one. The other solution includes the following simplified formulas:

2) $$\bar{a}_{ij} = \sum_{k=1}^{K} a_{ij}^{(k)} \frac{W_k}{N_a} \; ; \quad \bar{b}_{ik} = \sum_{k=1}^{K} b_{ik}^{(k)} \frac{W_k}{N_b} \; ; \quad \bar{\pi}_i = \sum_{k=1}^{K} \pi_i^{(k)} \frac{W_k}{N_\pi}$$

Rabiner [10] established that a good estimation for W_k could be $W_k=1/P_k$ where $P_k=P(kth\ training\ seq.\ |\ model\ for\ seq.k)$. Some other studies as presented in [9] include also as acceptable values $W_k=1$ or $W_k=P_{all,k}$ where $P_{all,k}= P(all\ training\ seq.\ |\ model\ for\ seq.k)$. To verify which procedure performs better, the appropriate research is conducted. Results are shown in table 8.

Table 8: Results of the system performance for known user according to different W_k values

		$W_k=1$	$W_k=1/Pk$	$W_k=P_{all,k}$
Method 1	Accuracy (%)	85%	60%	40%
	Training Time (m)	32	33	30
Method 2	Accuracy (%)	25%	15%	5%
	Training Time (m)	30	31	28

b) Multi-trainers: Training models by multiple speakers.

The previous extensive section investigated the influence of several variables in the training process using simplified models with a single speaker for recordings. It is time now to use the knowledge gained to develop models from recordings of different voices.

The first issue to determine is the number of speakers that should train the models. Reality matches the intuition: as many training voices, as better system performance. Powerful communication companies and big university projects follow this principle closely investing huge quantities of money and resources until they have a sufficient collection of samples to enable universal voice recognition. As a significant example, the corpus CSLU Alphadigit at Oregon Graduate Institute is a collection of about 78,000 examples from 3,031 talkers saying strings of letters and digits over the telephone. Some other institutions with more limited resources resort to a smaller number of samples to train their models, but they still pay hundreds of speakers to get correct recognition. As an interesting example, The Australian In-Car Speech Corpus has been built by utterance of 50 different speakers in different noisy conditions.

However, all these big technical deployments stay away from the goal of this thesis. The present thesis investigates the current voice control algorithms in order to implement a basic demonstrator which recognizes the maximum number of commands and speakers. The final number of voices was finally limited not only by the difficulty of finding availability; it was mostly because the required time and computational resources to process all operations increases exponentially with the number of speakers and the available resources were limited. As a reference, bear in mind that to process just the Codebook with 41082 samples corresponding to the recording of four different speakers it was necessary more than 28 hours of computation with the prepared resources.

Hence, after evaluating time requirements of each feasible option compared with possible future accuracy, a compromised solution was reached; the system would be trained by six different voices, 3 males and 3 males, who recorded 20 repetitions of each word in a quiet environment. Specifically, the sample group was composed trying to get variability of pronunciations and accents. Young and sportive people between 20 and 24 years with Spanish, French, Italian or Portuguese accent were selected. Swedish and native English accent were

highly recommended but, at the end, it was not possible to make an appointment with any speaker of this range. It will be a task for future system enhancements.

These recordings were useful to:

-Train the codebook: Due to the special difficulties to classify the large amount of data, it was decided to pick just the recording corresponding to the three most representative voices. Results show that when the number of chosen voice for the training increased to 4, the samples classification was flawed. See results of this training in table 9.

-Train the word models. Since the library of recording was built with the voices of 3 males and 3 females, the options to train and test the models are diversified.

Creating individual model for each individual repetition had not got any sense, since the associated number of repetitions associated to a word exceeded one hundred. But it was possible to try different models distributions to see which one got better performance.
Basically it was decided to make 3 divisions. They are presented below.

- **Common model:** Only one model was created per word. Each word model was trained by all the repetitions associated to that word, regardless of the speaker.
  9 models were built with this configuration.

- **Gender model:** Two different models were created per word, to have a male and a female model for word. One of them was trained by men voices and the other by women's. At the end, 18 models were built and trained. It was important to know if trained models differed quite much between different genders. If they do so, it will imply that classification of phonemes in the Codebook was made also according to gender, and therefore the system should be adapted to this speaker characteristic.
  In the recognition process what the system does is to estimate the probability of the input sequence to be generated by each of the gender models and the word related to the model with a higher likelihood will be accepted as the recognized.

- **Individual model:** 6 different models were built per word, as many as the number of library voices. In total 6x9word=54 models compose the library. The idea is the same than in the previous case; the model with higher probability to generate the observation sequence is the recognized one.
  It is predicted a considerable variability of results when the system is used by an external party, because the pronounced word can match with more models. For a known user, the recognition accuracy should be acceptable if the training model were correct.

Table 8 contains all the parameters and results for each experiment. Reported accuracy data are estimates of precision of the system according to different type of users.

Table 8: Estimated accuracy of the system as a function of then number and distribution of the trained models

Models Config.	Training models			Estimate accuracy of the system			
	Nº Total of Models	Average Nº of repetitions/word	Total time	Known user		External user	
				Male	Female	Male	Female
Common	9	170	56	70%	65%	30%	25%
Gender	18	85	58	90%	80%	65%	60%
Particular	54	28	140	85%	85%	45%	40%

4.3. Followed method

When at first the development of this master thesis was accepted although the general idea of the works was clear, the level of knowledge about voice control system and their algorithms was very poor. Although Java programming was familiar, there also was a totally lack of knowledge in Android applications and its operability.

The first task that was developed was a long and completed process of documentation in the library and through the Web, trying to find appropriated sources which could explain clearly how these systems work. The initial chosen sources were mostly too complex and abstract in concepts for a novice. Since it was impossible to get enough understanding from written sources, it was decided to start looking for an open and educational voice recognition system. Due to its relevance the Sphinx project from the Carnegie Mellon University was rapidly discovered. It is an educational project with a big collection of instructive papers and several open C and Java programs useful to understand and learn from zero how speech recognition works.

The learning process was slow, alternating the reading of papers with the study of the current software. The level of adaptation of the SphinxPocket software made possible after a very hard work to program an Android speech recognition application based on this software. However, finding out that speech recognition differed quite much from voice command recognition was the key strike that made the study goes forward. From that point, documentation searches were focused on Automatic Speech Recognition oriented to command recognition. It was known what it was necessary to do, but not how.

The conversion of the input audio into a matrix of features was the first difficulty to face. It was necessary a preliminary study of how the Android terminal gets and record audio. Then the implementation of the code for processing the signal was developed following different articles, tutorials and some open code examples.

When this signal processing technique was completed, easily the Dynamic Time Warping Technique for matching temporal vectors was developed. Correct results were gotten almost at the first trial, and several tests were successfully developed according to the level of performance that this method could offer.

As a reference in patter recognition all the documents were pointing the use of Hidden Markov Models. Despite of the wide use of this technique in multiple technologic applications, this method was totally unknown for this writer, so again a big effort was made trying to understand and assimilate its principles. Referred documentation usually mentions the use of observation vectors to train the models, but most of them omit explanations about how to get these vectors. Finding out this procedure extended the work around two weeks more, but once that the Vector Quantization Technique was studied, its implementation was not difficult at all since it is almost an elemental procedure to quantify vectors by numbers. Besides, some developed open codes were taken as a reference. The VQ procedure was developed using two main algorithms: K-Means and LBG. The implementation of the second one was fast and apparently their results were correct. But after working normally with it for more than one month, farther results in HMM training showed the shortcomings of this method.

The extensive information about the HMM problems made the implementation of the corresponding algorithms easier. The though initial point was to train models individual models according just to one observation sequence. The process until success was long due to natural difficulties of the method. Inconsistencies in results made the programmer review and reprogramme the code several times. It was even necessary to get an alternative approximation of the technique in order to avoid the system to work with infinite values. All this researching process adapting the theory to the code was a nice process which makes this writer feels proud of the engineering work.

Once that model for unique sequences was developed it was missing a procedure to train this model according to different repetitions. Once more, it was necessary a depth study in the field to discover the specialized procedures for this cases: multiple sequence training. Mainly read papers where always referring to Rabiner's procedure in this sense, so his idea was taken to resolve the problem. At the end of this phase, the application was successful creating and training properly models for several repetitions from one speaker. As a consequence, the recognition of commands said by this speaker has also a high rate of accuracy, over 90%.

To extend the recognition to more users was a primordial objective, and only with one trained voice the accuracy was not good enough. Due to that fact it was decided to extend the number of voices that train the system finding a compromised solution between accuracy and time to process the info. Since the project was carried with some delay, and lessons had already finished it was difficult to find all the desirable voices. Nonetheless, at the end a mixed and representative sample of voice were recorded.

With all these voices the system was trained in the three explained ways and results were evaluated by testing the program with different voices. Time data was easily measurable, but the measures about accuracy are more estimates than precise information since it is not available either a considerable group of testers.

Regarding to the communication between terminal and radio, since the beginning the objective was trying to find the most portable and comfortable solution. The first idea was to use the phone as the host terminal to control the communication. It was known that although

some similar models initially do not accept this host capability, it could be possible to introduce some modifications in its kernel. Documentation and tutorials were widely analyzed and finally it was discovered that this specific model lacks this capability. As an alternative and as the cheapest solution the use of a laptop to control the communication could be an acceptable solution. Therefore, the most portable solution was developed with that limitation. A wireless (TCP/IP) communication between PC and mobile phone was carried. The communication between radio and PC had to be via serial communication. Both systems were individually implemented fast and effortlessly linked. However in the test of the system a flaw was detected in the PC-radio communication. The problem was not in the software; some hardware component was failing and the problem has not been found so far.

Chapter 5

Conclusions & further discussions

Throughout this document it has been presented the theoretical study in voice control algorithms and its subsequent practical implementation in an operative *demonstrator*. The way to achieve the presented results has been long and tedious, with an extensive learning process. Although many setbacks have hampered the work, finally the work has succeeded fulfilling the objectives of the thesis.

Throughout the previous sections, the followed method with brief comments on the obtained results has been presented in an objective way. The aim of this chapter is first to describe the used method in order to later make a critical analysis of it, evaluating its implementation and effectiveness. Then based on this analysis, possible improvements and future related works will be described. At the end recommendations about how to understand and use the system will be exposed.

5.1 Method, development and success

The exposed method in the previous section can finally be consider as successful, but it can result too long and complex for the real level of difficulty involved in the project.

One of the main factors which interfered in the fast development of the project was the initial confusion between Automatic Speech Recognition (ASR) and Automatic Speech Understanding (ASU). This fact induces partial waste of time trying to understand the basis and

principles of a much more advance technique which it was not required. If since the beginning the study of documentation were focused on this sense, it could have been possible to save at least one month of work, avoiding previous unnecessary studies and software development as the one related with the Sphinx project.

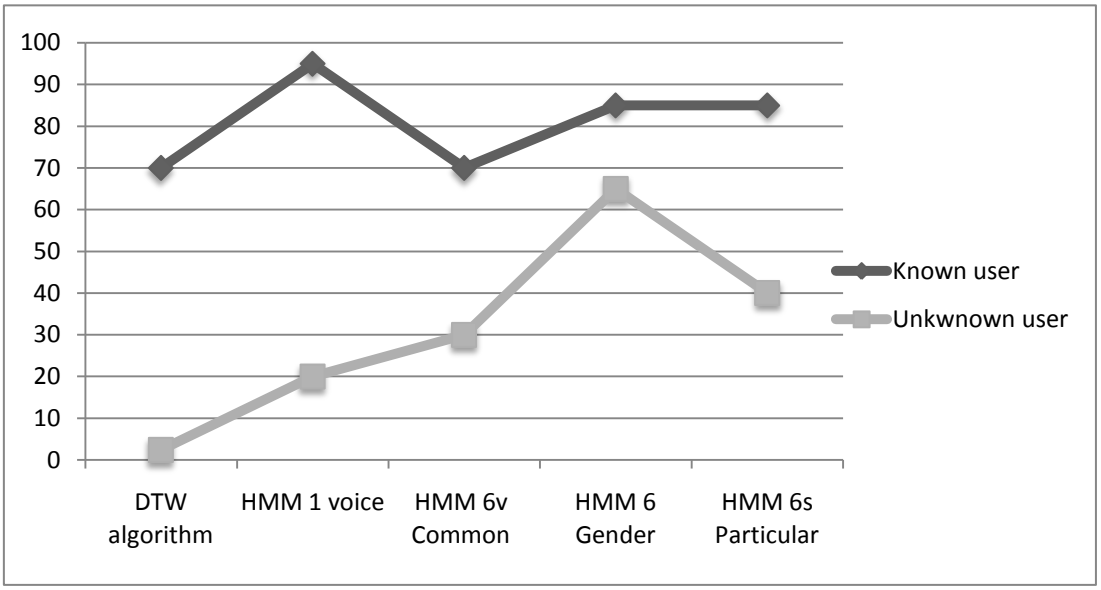
Once the right way was driven, algorithms implementation for signal processing was easily developed thanks to good documentation on this process made in advance. Although it may seem not effective for multi-voices, DTW technique implementation can be considers totally valid when it works with trained voices. The named technique archives good comparisons between two temporal series, but only good matches are found when series are similar in time and also their characteristics are similar, or what it is to say in this case, when they came from the same source.

Satisfied with the work but not with the intrinsic results of the studied algorithms, the work was moved forward trying to get better results in processing time and accuracy in recognition. The development of the HMM technique was complicated by the huge amount of information and documentation about it. Move from one paper to another trying to find fast solutions instead of studying properly a good manual of reference, at the end was a considerable mistake which was paid with time and several subsequent mistakes. The correct information was going extracted slowly and slowly implemented but always checking each step. However, all these confusions were with patience overcome, finding the appropriate papers. Brilliant results for individual recognition were obtained.

In order to improve the accuracy of the system for any kind of users it was decided to start from 6 different voices which would be recorded. The recording process always followed the same procedure although it was tried it to introduce some variations in the pronunciation of each speaker in order to get higher variety of sounds. The read information about how to train models from different voices did not offer almost any alternative and it is here where the student's reasoning plays a decisive role. Three main studies were proposed and developed in order to find out an optimal solution. Finally, after the study and evaluation of the result the chosen technique offers acceptable results.

A clear evolution in system efficiency and accuracy has been presented though this project. The developed method was thought in order to make step by step breakthroughs in the research. The first results came from basic systems with low accuracy by implemented the DTW technique. Nonconformity with the archived results, ambition and enthusiasm motivated a broad study and analysis in HMM algorithms which made possible improve the efficiency and accuracy of the system for a known user. Expand command recognition from one trained voice recognition to universal voice recognition was the huge challenge of the project. By proposing three different approaches, it was possible to combine and manage different alternatives, which finally was decisive to get these good results.

Evolution in the accuracy of the system for known user and universal recognition is clearly visible in the graph 1. In x-axis the implemented algorithms are present in chronological order. Efficiency of each algorithm is rated in percentage in y-axis.



Graph 1: Evolution in the accuracy of the system for known and unknown users

From the graph it is possible to get interesting conclusions:

·The efficiency of DTW algorithm is acceptable for a known user, but HMM supposes a big breakthrough in the accuracy of the recognition raising the mark up to 90%. Besides this HMM 1 voice allows for first time unknown user recognition.

·Adding more trainer voices increases the success rate in universal speaker commands recognition. The choice of the algorithm in this regard is essential to obtain acceptable results. The effect of more voices on the recognition of trained voices is understandably accused, but is less significant in the model by genders.

·Universal recognition rates increase with the number of added voices, but they generally remain in low rates, below 50%, far removed from those obtained with familiar voices. However, it is precisely in this area where the positive development of the project is more visible, giving an improved performance with each new voice.

The implementation of an algorithm based on the distinction of male and female voices is a great success. It increases the universal recognition up to acceptable values and keeps the rate of familiar voices recognition around 90%. This is, therefore, the final chosen model.

·Implement individual models for each voice in the two types of recognition decreases the accuracy of the system. This is because many models interacting in the final decision have a negative effect on the results. That is to say, the spoken word may seem to pair with many different models, which due to the large variability of pronunciations induce errors.

·Working with a common model for all voices introduces distortion into the system from trying to find the optimal combination between high variability of voices without even distinguish their gender.

5.2 Possible improvements and future researches

The primary objective of this work is to recognize voice commands from multiple speakers in the most accurate and efficient manner. To archive it, a working method has been designed and developed; it improves and choices the optimal configuration of parameters in each stage, so that ultimately the obtained system has the highest values of precise tolerance with minimum cost in resources.

As it was shown in the previous section, the optimal working point of the system is archived with an HMM model by genders. In this configuration, the hit rate for unknown voices is around 65% while hit rate for trained voices is remained around 90%. These figures are positive taking into account the great difficulty of this task. It has been achieved a great efficiency with limitations in time and resources, with a reduced number of trained models and implementing the most efficient system saving extra features. It was a difficult task for an inexperience and autonomous student.

Despite the good feelings left by this research university work, note that any command recognition software you want to succeed in the market should have a hit rate close to 95 percent for all types of voices. Comparing this desirable rate with the obtained 65% for unknown voices it is understandable how new projects and research priorities should improve success in universal speaker command recognition.

The development of this system and the research work done on this and other systems allows some rabbits and suggestions in order to improve even further the precision and efficiency of future systems. First of all, it is basic to assume that the best way to improve the accuracy and efficiency of the whole system is to improve the implementation and efficiency of each subprocess and sub-task. Focus on this principle; a completed list of areas and process is presented, with some suggestion and advices to get better performance:

·**Recording system and frames obtainment:** The implemented method uses a sample rate of 8000 Hz without overlap. Each frame has 256 samples, and therefore each frame is gotten each 32 msec. Although this rate was chosen based on specialized literature, it is high recommendable to develop a system with overlap, testing different sample rates and frames size. 32 msec might not be the optimal average period to represent a phoneme.

·**Filtering and noise cancellation:** No additional noise reduction technique has been applied in the system apart from the mel-filters applied in input spectrum. Besides, it may be possible that the terminal originally includes some filters to reduce noise in the recording process. However, this might not be necessary and therefore implementing any noise reduction technique could be very beneficial. Noise Reduction Filtering (NRF) and Noise Cancellation (NC) techniques will reduce and avoid the presence of ambient noises in the final recorded audio signal.

·**MFCC Coefficients:** They are the most popular ways to represent a sound since they result from the application of Computing Discrete Cosine Transform over log mel filter-bank energies. However, the robustness of this procedure is questionable in the presence of additive noise, even if their values are normalized. Some papers propose modifications to the basic MFCC algorithm to improve robustness. See references [3], [11], [12]

Besides, the consulted literature encourages working with 39 MFCC instead of 13 to have also a representation of how these coefficients change (speed and acceleration). As it is detailed in page 20, working with 39 coefficients could be very advantageous since it permits more accurately to distinguish similar sounds that differ in their dynamics. However with 39 MFCC the system does not provide accurate results but the specific reason remains still unknown. It could be very interesting to find out the reason, and implement a more completed system with all the coefficients.

Some others filter as the Bark Filter Bank could be also attractive to investigated to see if they produce similar coefficients with higher accuracy.

·**Obtaining the observation vector**: Once that an audio input is recorded and transformed into a matrix of coefficients, it is necessary to transform this matrix into a vector, quantization the arrows of the matrix, classifying each frame (vector of the matrix) by the sound that it represents. A proper implementation of this procedure is essential to ensure the correct classification of sounds and subsequently the future success of the recognition system. The implemented method is the Vector Quantization Technique through the K-Means algorithm. This implementation apparently performs well, but it is not the most efficient tacking a huge part of the training stage. The LGB algorithm was also developed but the obtained results were not expected, they could not satisfy a correct classification. An important objective for a future work could be focus on implementing the LGB algorithm in a different way which ensures efficiency and better performance.

In the same manner, it could be also interesting to investigate some other ways to classify and quantify different sounds. Possible methods using eigenvalues and eigenvector could be study and became part of future thesis and others researches.

·**Models training:** In the HMM technique, having accurate models which suitably represent each command word is essential to guarantee the accuracy in the recognition. Rabiner [13] describe how to train the models and which are the main parameters which influences in the accuracy of the models.

> ·**Larger number of speakers:** As it is justified in page 48, in this work it was impossible to get more than 3 couple of different nationalities with different accents, so in total there were just 6 voices to train the HMM models. Therefore, the variability represented in the models after the training was very reduced if we want to perform universal voice command recognition. It is vital for new versions of the software to extend the number of voices and study how this extension affects to the global accuracy. The universal system accuracy is expected to rise to 90% just augmenting the number of voices until 50 voices. This must be a priority for future researches, since it will lead to more pronounced improvements.
>
> ·**Larger number of repetitions**: The number of repetitions per speaker and word was set at 30, trying to reach a compromise solution to get a good workout without taking too prolonged time. If suggestions related with time saving were implanted, it would be possible to reduce the training time, and subsequently increase the number of repetitions to get better global performance.

·**Models selection and training:** The main faced problem in training model was not how to train a model, either which algorithm was recommendable to follow. The main raised uncertainty was how to combine different voices and repetitions to try the models in a manner that could result optimal providing the best results. In this sense, as described on pages 49, 3 types of studies were conducted. However, new and more distant reasoning and research in this regard may lead to new modes and settings with better results.

·**Estimates:** the estimates made in the various on the accuracy of the system tests were conducted according to the formulas presented in section 4.2.1. These formulas allow few testers using the system, generalizing the results and getting a good indication of system performance, but playing with a high level of uncertainty. However, an increasing number of people testing the system will reduce the level of uncertainty and obtain more accurate load.

The acquired knowledge and the done work allow future versions, which can easily expand the number of command and languages accepted by the system. With proper training of the models, it could be even archived the recognition of any acoustic pattern.

5.3 Recommendations

The use of the different Android applications is intuitive. It is not necessary extra information about this issue. Regardless, there are some advice that should be reported.

The sensibility of the system is quite high in order to make the recognition available for the maximum number of users. That means that the system should be careful treated or false commands will be identified. If the device is moved or crashed generating strange sounds it cannot be guaranteed that they are going to be omitted. For the same reason, it is highly recommendable to pronounce the commands as clearly as possible and in an intermediate-high voice volume.

The unit is designed for its use outdoors, so is advisable to avoid any use in noisy environments, and if it is so, the sensibility of the system should be reduced. At the same time, it could be good to maintain the terminal no far away from the mouth, or in a more advanced case using headphones with microphone.

Based on the results and in order to ensure good rates in the recognition, it is highly recommended to train the system with the own voice.

Bibliography

[1] D. J. &. J. H. Martin, *Speech and Language Processing: An introduction to natural language processing computational linguistics, and speech recognition.*, Second Edition ed. Publisher: Prentice Hall, 2009, ISBN-10: 0131873210.

[2] C. M. University. (2012, Jun.) CMU Sphinx-Open Source Toolkit For Speech Recognition. [Online]. http://cmusphinx.sourceforge.net/

[3] Md. Rashidul Hasan, Mustafa Jamil, Md. Golam Rabbani Md. Saifur Rahman, "Speaker Identification Using Mel Frequency Cepstral Coefficients," in *3rd International Conference on Electrical & Computer Engineering*, Dec. 2004, p. 4.

[4] S.B. Davis, and P. Mermelstein, "IEEE Transactions on Acoustics, Speech, and Signal Processing, 28(4)," in *IEEE Transactions on Acoustics, Speech, and Signal Processing, 28(4)*, 1980, p. 357–366.

[5] A. B. a. R. M. G. Y. Linde, " An algorithm for vector quantizer design," IEEE Transactions on Communications, 28(1):84-95, January 1980.

[6] Sakoe, H., Chiba, S., "Dynamic programming algorithm optimization for spoken word recognition," Nippon Electric Company, Limited, 0096-3518, 1978.

[7] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, *Spoken Language Processing; A guide to Theory, Algorithm and System Development*. Saddle River: Prentiee Hall PTR; Carnegie Mellon University;, 2001.

[8] I. Icom. UHF TRANSCEIVER ID-31 E Advanced Intructions. [Online]. http://www.icomamerica.com/en/downloads/DownloadDetails.aspx?Document=567

[9] Davis, Richard I. A. Lovell, Brian C. Caelli, Terry, "Improved Estimation of Hidden Markov Model Parameters from Multiple Observation Sequences," The Institute of Electrical and Electronics Engineers 0-7695-1699-8, 2002.

[10] Rabiner, Lawrence R. Juang, Biing-Hwang, *Fundamentals of Speech Recognition*. PTR Prentice Hall, 1993, ISBN: 0130151572, 9780130151575.

[11] D. Jurafsky, "Speech Recognition, Synthesis, and Dialogue. Lecture 9: Feature Extraction and start of Acoustic Model," p. 59, 2010.

[12] M. L. Seltzer, "Microphone Array Processing for Robust Speech Recognition," PhD Thesis, Carnegie Mellon University, Pittsburgh, PA 15213, 2003.

[13] M. Nilsson, "Speaker Verification in Java," School of Microelectronic Engineering Griffith University, October 2001.

[14] M. Murphy, *Android Programming Tutorials*, Third Edition ed. CommonsWare, LLC, 2011.

[15] Willie Walker, Paul Lamere, Lamere, Philip Kwok, Bhiksha Raj, "Sphinx-4: A Flexible Open Source Framework for Speech Recognition," SUN MICROSYSTEMS INC. SMLI TR2004-0811, 2004.

[16] C. Fang, "From Dynamic Time Warping (DTW) to Hidden Markov Model (HMM)," University of Cincinnati, Final project report for ECE742 Stochastic Decision, 2003.

[17] G. S. Eberhard Hänsler, *Speech and Audio Processing in Adverse Environments*, illustrated ed. Sep 17, 2008, 3540706011, 9783540706014.

[18] C. U. E. Department, *The HTK Book*, 34th ed., C. University, Ed. 2006, http://htk.eng.cam.ac.uk/docs/docs.shtml.

[19] M. Rabani, "Information Bottleneck and HMM," The Hebrew University of Jerusalem 031487309, 2006.

[20] S. Mangayyagari, "Voice recognition system based on intra-modal fusion and accent classification," University of South Florida Theses and Dissertations; Paper 2274, 2007.

[21] J.-P. Hosom. (c, Jun.) Center for Spoken Language Understanding. [Online]. http://www.cslu.ogi.edu/

[22] J. Choi, "An FPGA implementation of speech recognition with weighted finite state transducers," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, Sch. of Electr. Eng., Seoul Nat. Univ., Seoul, South Korea, 14-19 March 2010, pp. 1602-1605.

[23] R. Bellman, *Dynamic Programming*. Courier Dover Publications-2003, 1957, ISBN-10: 0486428095, ISBN-13: 9780486428093.

[24] Arthur Chan, Evandro Gouvea, Rita Singh, *The Hieroglyphs: Building Speech Applications Using CMU Sphinx and Related Resources*. March 11, 2007, Third Draft.

[25] R. H. Christopher Tarnas, "Reduced space hidden Markov model training," University of California, 1998.

[26] S. E. B.-G. a. A. O. Asadi, "Hans-free Voice Activation Personal Communication Devices," Conexant Systems, Inc. Media Access, Speech Technology Development.

[27] A. K. N. B. Anjali Bala, "Voice Command Recognition System based on MFCC and DTW," *International Journal of Engineering Science and Technology . Kurukshetra University*, p. 8, 2010.

[28] R. K. Amin Atrash, "Development and Validation of a Robust Speech Interface for Improved Human-Robot Interaction," *International Journal of Social Robotics*, no. DOI 10.1007/s12369-009-0032-4, p. 14, Sep. 2009.

[29] Ganesh Tiwari, MadhavPandey, ManojShrestha. Supervisor : Dr. Subarna Shakya. Major Project Mid-Term Presentation :Speaker Verification for Remote Authentication.

[30] M. Ordowski, N. Deshmukh, A. Ganapathiraju, J. Hamaker, and J. Picone, "A public domain Speech-to-Text System," *Mississippi State University, USA*, p. 8.

[31] B. R. a. R. M. S. Michael L. Seltzer, "Speech Recognizer-Based microphone array processing for robust-hands free Speech Recognition," Carnegie Mellon University, Mitsubishi Electric Research Labs.

[32] A. T. Mahdi Shaneh, "Voice Command Recognition System Based on Based on," *World Academy of Science, Engineering and Technology 57*, p. 5, 2009.

[33] M. B. B. W. Ian Booth, "Enhancements to DTW and VQ decision algorithms for speaker recognition," Speaker Verification Group, Department of Electrical Engineering, University of Queensland, May 1993.

[34] E- Hocine Bourouba, Mouldi Bedda and Rafik Djemili, "Isolated Words Recognition System Based on Hybrid Approach DTW/GHMM," Department of electronic Faculty of Engineering, University of Annaba, 2005.

[35] D. A. Reynolds, "An Overview of Automatic Speaker Recognition Technology," MIT Lincoln Laboratory.

[36] M. Stamp, "A Revealing Introduction to Hidden Markov Models," Department of Computer Science; San Jose State University, April 26, 2012.

[37] R. Singh, "Designing HMM-based ASR systems," *School of Computer Science, Carnegie Mellon University, USA*, 2003.

[38] Y. S. Naous, G. F. Choueiter, M. I. Ohannessian, M. A. Al-Alaoui, "Speech Recognition for Voice Based Control," Faculty of Engineering and Architecture A.U .Beirut.

[39] Lindasalwa Muda, Mumtaj Begam and I. Elamvazuthi, "Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques," *Journal of Computing*, p. 6, March 2010.

[40] L. R. Rabiner, "A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *IEEE*, February 1989.

[41] Z. G. S. Z. Zheng Fang, "Comparison of Different Implementations of MFCC," Department of Computer Science and Technology, Tsinghua University Vol.16 No.6 J. Comput. Sci. & Technol., February 23, 2001.

Appendix A

List of implemented programs in Android with their corresponding description

·*RecorderCB*: 48 KB Application through which the recording is performance. It counts with 9 bottoms associated to the recording of each word. When the each recording is finished, the program processes the audio and converts the recording into a matrix of features.

·*PC-Trainer:* It is a Java program which implements the training of any Codebook by adapting different parameters.

v1.0 Implement the standard K-Means algorithm successfully.

v1.0.1 Working with different number of clusters
v1.0.2 Working with different number of repetitions

v1.1 Tries to gain comfort and speed through the LBG technique.

·*HMM_Trainer:* There are three different versions of this android application. All of them are aimed at training the HMM models according to which word repetitions they use.

·*HMM_Trainer_Mono:* Trains the model with only one speaker's voice.

v1.0 Working with models of 6 states
v1.1 Working with models of an adapted number of states

·*HMM_Trainer_Common:* It trains one model per word using all the available voices.

·*HMM_MultiModel:* Trains as many models per word as numbers of available voices.

·*HMM_Trainer_Gender*: It trains two models per word according to speakers' gender.

v 1.0 Working with Wk in the first multimodel training
v.1.1 Working with different Wk in the second multimodel training

·*VoiceCtrl_DTW:* Android Program of command recognition through the Dynamic Time Warping algorithm. As all the recognition applications, it counts with one bottom to start the recognition and another to stop it. The rest of the control is made by voice.

v1.0 Working with 13 MFCC

v1.1 Working with 39 MFCC

·*VoiceCrtl_HMM*: Android application which performs the voice recognition after all the HMMs are trained. There are four versions of this application, each one prepared to work according to the correspondent HMM trainer.

·*VoiceCrtl_HMM _Mono:* Recognition using only one speaker's voice training.

·*VoiceCrtl_HMM _Common:* Recognition with all voices in one model/word.

·*VoiceCrtl_HMM _MultiModel:* Recognition with multimodels.

·*VoiceCrtl_HMM _Trainer_Gender:* Recognition with models by trainers' gender.

·*RadioCom*: Program in Java which runs in the PC. It controls the communication between the radio and the Android Terminal.

·*VoiceCtrl_Sphinx:* It was the first programmed application for Android. It is particular adaptation of the Sphinx Software for Android which performs speech recognition. The original code was widely studied trying to understand the principles of a voice recognition application.

Appendix B

Pseudo Code of different algorithms for Voice Recognition

B.1 Pseudo Code for the K-Means algorithm implementation

Let

N=space dimension. Length of vector and cluster
O=number of observation vectors
MaxIter= maximum number of iterations

Input

E = {e_1,e_2…, e_O} = set of vectors to be clustered.
e_i= vector of length N
K=number of Clusters

Output

C= {c_1,c_2…, c_k} = set of cluster centroids
c_i=centroid of the cluster i

1. Initialization

Create initial centroids

for *k=0* to *K-1*
    for *i=0* to *N-1*

$$Coordenates\_c_k[i] = \left(\left(\frac{V.MaxCoordenates[i] - V.MinCoordenates[i]}{k+1} \times n \right) + V.MinCoordenates[i] \right)$$

    next *i*
    set centroid k
next *k*

Assign each observation vector to the closest centroid

min_d=inf

for *i=0* to i=*O-1*
    for *j=0* to *j=K-1*
    eu_d= Euclidean Distance (e_i,c_j)
    if (eu<md)
        min_d=eu_d
        assign e_i to c_j
    next j
next i

TotalDesviation= calc_Total_Desviation
TotalDesviation'= 0

2. Iteration

*While ((Total_Desviation'<Total_Desviation*0.999) & (Cont_Iter<Max_Iter))*

Find the centroid of the cluster among all the vectors assigned to the Cluster

Total_Desviation= calc_Total_Desviation

for *i=0* to *i=K-1*

for *j=0* to *N-1*
sum=0;
for *h=0* to *Number Of Vectors in Cluster i*
sum=sum+e_h[j]
next h

$cluster_i$[j]=sum/h
next *j*

next *i*

Assign again each observation vector to the closest centroid

Total_Desviation'= calc_Total_Desviation
Cont_Iter++;

Return final C= {c_1,c_2…, c_k}

B.2 Pseudo Code for HMM Problem 3. It also includes solution to P1 and P2

Let

N=number of states MaxIter= maximum number of iteration

M= different type of observations

T=number of observations

1. Initialization

Select initial values for the matrices A, B and π

See initial expression in page 48

n_iters=0

old _prob=0

2. α computation

//compute $\alpha_0(i)$

$c_0 = 0$

for *i=0* to *N-1*

$\alpha_0(i) = \pi(i)b_i(O_0)$

$c_0 = c_0 + \alpha_0(i)$

next *i*

//compute $\alpha_T(i)$

for *t=1* to *T-1*

$c_T = 0$

for *i=0* to *N-1*

$\alpha_T(i) = 0$

for *j=0* to *N-1*

$\alpha_t(i) = \alpha_t(i) + \alpha_{t-1}(j)a_{ji}$

next j

$\alpha_t(i) = \alpha_t(i)b_i(O_t)$

$c_t = c_t + \alpha_t(i)$

next *i*

next *t*

3. β computation

//compute $\beta_{T-1}(i)$

$\beta_{T-1}(i) = 1$

//compute β

for *t=T-2* to *0* by *-1*

for *i=0* to *N-1*

$\beta_t(i) = 0$

for *j=0* to *N-1*

$\beta_t(i) = \beta_t(i) + a_{ij}b_j(O_{t+1})\beta_{t+1}(j)$

next *j*

next *i*

next *t*

4. $\gamma(i, j)$ computation

for *t=0* to *T-2*

den=0

for *i=0* to *N-1*

for *j=0* to *N-1*

den=den+ $\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)$

next *j*

next *i*

for *i=0* to *N-1*

$\gamma_t(i) = 0$

for *j=0* to *N-1*

$\gamma(i, j) = (\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)) / den$

$\gamma_t(i) = \gamma_t(i) + \gamma(i, j)$

next *j*

next *i*
next *t*

5. Re-estimate A, B and π

//re-estimate π

for *i=0* to *N-1*
$\pi_i = \gamma_0(i)$
next *i*

//re-estimate A
for *i=0* to *N-1*
for *j=0* to *N-1*
num=0 den=0
for *t=0* to *T-2*
num=num+ $\gamma(i, j)$
den=den+ $\gamma_t(i)$
next *t*
a_{ij}*=num/den*
next *j*
next i

//re-estimate B
for *i=0* to *N-1*
for *j=0* to *M-1*
num=0 den=0
for *t=0* to *T-2*
if (O_t*=j)* then
num=num+ $\gamma_t(i)$
end if
den=den+ $\gamma_t(i)$
next *t*
b_{ij}*=num/den*
next *j*
next i

6. Calculate P(O|λ)

for *i=0* to *N-1*
Prob=Prob+ $\alpha_t(i)$

next *i*

7. Iteration

If (prob'>old_prob) & (iter<MaxIter)→ Iterate