MASTER THESIS

TITLE: Start-and-End Point Detection at the Input of Speech Recognition Application

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Nazaret García Trascasa

SUPERVISOR: Petr Pollak (CTU, Prague);

TUTOR: Francesc Tarrès (UPC);

DATE: June 2nd 2013

Resumen

Este documento tiene por objetivo recoger la información relativa al proyecto sobre la creación de un algoritmo para Start-and-End point detection de una señal pregrabada.

La intención inicial del desarrollo de este algoritmo es que pueda ser utilizado en la entrada de una aplicación de reconocimiento de voz. En términos generales, el resultado de este trabajo es un algoritmo que puede detectar el comienzo y el fin de una señal previamente grabada basado en un algoritmo de detección de la actividad de la voz previamente desarrollado por la Czech Technical University, Faculty of Electrical Engineering.

Hay dos temas principales de estudio en este proyecto: detección de la actividad de la voz (VAD algorithm) y determinar el punto de inicio y fin de la señal (Start-and-End point detection). El primer paso para la construcción del algoritmo final es ser capaz de identificar la actividad de la voz en una señal mediante el VAD algorithm para después ser capaz de detectar el inicio y final de la actividad de la voz y descartar los silencios de la señal mediante el Start-and-End point detection algorithm.

Con el fin de demostrar el modo de funcionamiento de dicho algoritmo se ha creado una aplicación en MATLAB que permite ver gráficamente una señal previamente grabada y posteriormente su punto inicial y final después de aplicar los algoritmos.

Por último, para proporcionar resultados más gráficos y dar al proyecto un valor añadido y con vistas a convertirse en una futura aplicación posible se ha añadido el reconocimiento de dígitos basado en de un algoritmo DTW (Dinamic Time Warping).

Overview

This document collects information on the proposed creation of an algorithm for Start-and-End point detection of a pre-recorded signal.

The initial reason for developing this algorithm is so it can be used at the input of a voice recognition application. Overall, the result of this work is an algorithm that can detect the beginning and end of a previously recorded signal based on a detection algorithm of the voice activity previously developed by the Czech Technical University, Faculty of Electrical Engineering.

Two main issues are studied in this project: Detecting the Voice Activity (VAD algorithm) and determining the start and end point of the signal (Start-and-End point detection). The first step to develop the final algorithm is being able to identify the voice activity in an audio signal by the VAD algorithm. The next step is to detect the beginning and end of activity and silence suppression by Start-and-End point detection algorithm.

To demonstrate the mode of operation of the algorithm, I have created an application in MATLAB to show graphically the process for a previously recorded signal and then the start and end points after applying the algorithms.

Finally, to provide better graphic performance and provide added value to the project, I have added a digit recognition algorithm based on a DTW (Dynamic Time Warping).

INDEX

List of Figures

List of Tables

INTRODUCTION

Speech processing is present in many applications of our day. It is an element in our daily lives of which we are often unaware. It is present in areas as useful for us as mobile communications, in which many advances have been developed that allows compressing the digital waveform representation of speech into a lower bit-rate representation, i.e., coding or speech compression. The mobile telephony field is seeing increasing use of smartphones, in which text-to-speech synthesis or voice-to-text is integrated into many current applications. More achievements are taking place in echo, noise or reverberation suppression, and speech recognition to make voice communication from human to human more realistic.

This report recounts the development of an application for voice detection in Matlab, exploring all aspects that relate to the project.

The starting point of this project is speech processing. The fundamental purpose of communication is the transmission of messages. Any message can be encoded as a set of bits in a waveform that can be treated to transmit, record, manipulate, and in the latter case, decoded by a receiver.

The achievements of the speech recognition field have been concentrated in different fields, including voice control of devices and transcription systems. For optimum results, it is necessary to process the audio signal. This processing can be performed by different algorithms. A voice recognition algorithm consists of several stages, including feature extraction and pattern recognition. In feature extraction, presented algorithms are best zero crossing rate, permanent frequency, cepstrum coefficient and liner prediction coefficients. Through a combination of these, we have developed the algorithm that we will use throughout the project: Voice Activity Detection.

VAD algorithm can detect when there has been voice activity and once voice is detected, the silences will be deleted. The work for this project involved taking an initial algorithm as a starting point and making improvements to optimize it, according to the type of audio signals that the algorithm will work with to obtain best results. Once it has detected voice activity in a given signal, the silences were deleted based on the end point detection algorithm, resulting in a signal without absence of voice to be stored in a buffer. Finally, the new signal is passed to the speech signal recognition and sets up voice patterns for comparison with the signal we want to process. Through comparison with these patterns, the application is able to return results in a written sequence of the audio signal.

The purpose of this project is to separate the active parts of the voice from an audio signal and then to establish recognition of known patterns. The objective is to acquire a deeper understanding of signal processing and speech recognition. On the other hand, the work requires a good understanding of the principles the VAD algorithm is based on and how to apply these concepts to achieve optimal results. Finally, the method for the recognition of patterns must be identified to make the full application work.

The last issue that we address in this introduction is the organization of this document. First, the paper will examine several theoretical concepts that

are essential for understanding the development of the project. This discussion will be divided into three main sections. The first will explain the basics of speech signals. Second, it will analyze the basics of speech processing signals that are relevant to this project. Finally, it will discuss speech recognition concepts and models that will be applied later to the algorithm.

The second chapter will discuss the Voice Activity Detection algorithm. First theoretical, issues will be addressed, including the cepstral analysis and its applications and the general principles of the algorithm. Then, it will discuss Start-End point detection, reveal the algorithm developed for the project, and explain its mode of operation.

The third chapter will explain how the development and integration of these algorithms. It will also show how it works.

Finally, the report will illustrate the operation of the application, the testing phase that has been carried out to check for proper operation, and the results obtained.

CHAPTER 1. THEORETICAL ASPECTS OF SPEECH PROCESSING

This chapter will describe theoretical aspects related to subsequent investigations. It is important to know how audio signals are produced to treat them in the proper manner.

This chapter will first explore concepts of speech signal, including how these signals are produced, their technical characteristics, and models of speech production. The next section will explain concepts of speech signal processing and their current applications in different fields. Finally, it will generally describe speech recognition and a linear predictive coding (LPC) model.

1.1. Basic Concepts of Speech Signal

This chapter intends to discuss how the speech signal is produced and perceived by humans. This first chapter is an introductory section that must be considered before one can understand what strategy is followed for voice recognition.

1.1.1. Speech Production

Speech sounds are produced when air from the lungs passes first over the glottis and then out of a person's throat and mouth. Depending on the sound articulate speech, the speech signal can be excited in three possible ways [1].

- Voiced excitation: Air pressure forces the glottis to open and close periodically, generating a periodic pulse train (in a triangle). This "fundamental frequency" is generally in the range of 80Hz to 350Hz.

- Unvoiced excitation: Keeping the glottis open, air passes into a narrow passage in the throat or mouth. This results in turbulence that generates a noise signal. The noise spectral shape is determined by the location of the stricture.

- Transient excitation: A temporary closing of the throat or mouth will increase the air pressure. When the closure is opened suddenly, the air pressure drops immediately ("blast blast").

In most cases, the emission of sound results from a combination of these three type s of excitation. The spectral shape of the voice signal is determined by the shape of the voice tracks (the tube formed by the throat, tongue, teeth, and lips). Changing the shape of the pipe (and also opening and closing the airflow through the nose) changes the spectral shape of the voice signal, so that different speech sounds are articulated. The following chart shows the process of speech production.

It is important to know how to produce audio signals then can treat them in the proper manner. This is the reason why in this chapter are being exposed to the basic principles of speech signal.

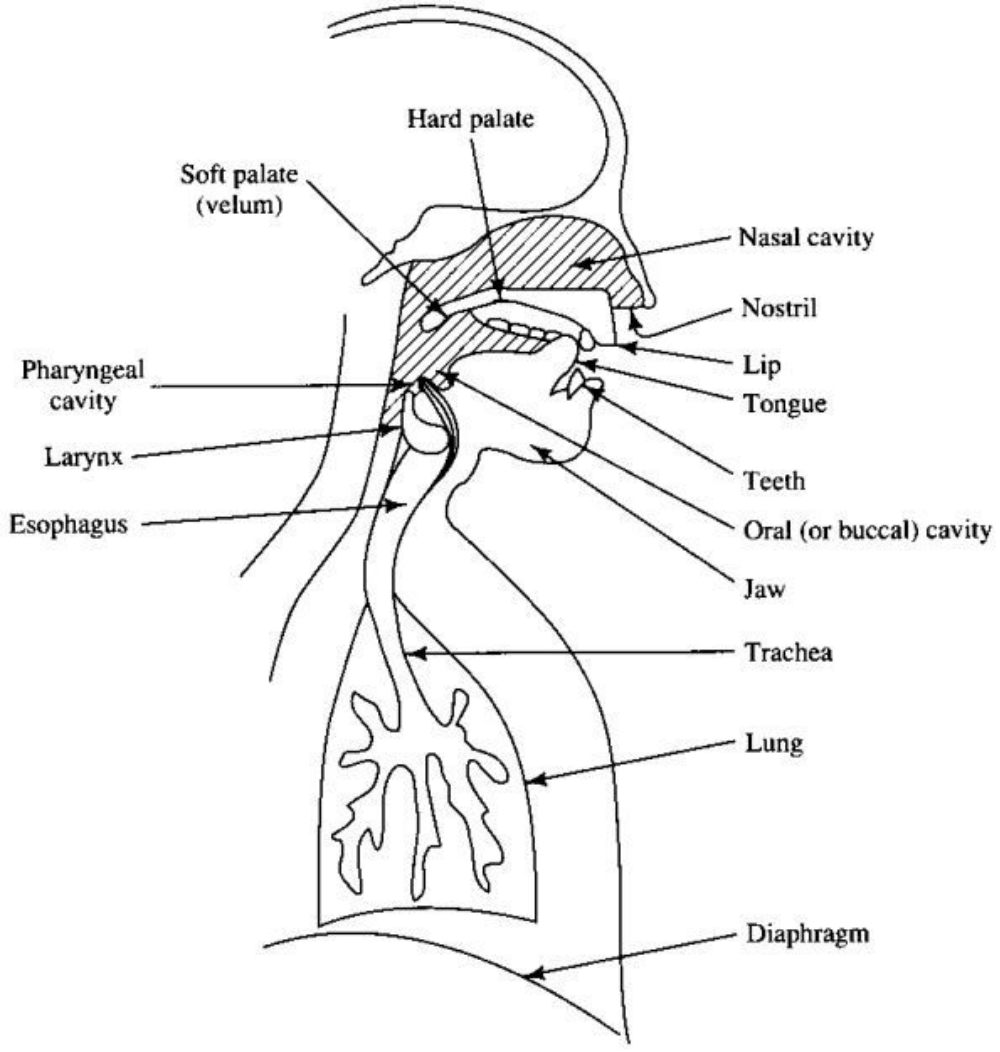


Figure 1. Speech production.

1.1.2. Simple Model for Speech Production Signals

The production of speech can be separated into two parts: producing the excitation signal and forming the spectral shape. Figure 2 shows a simplified model of speech production.

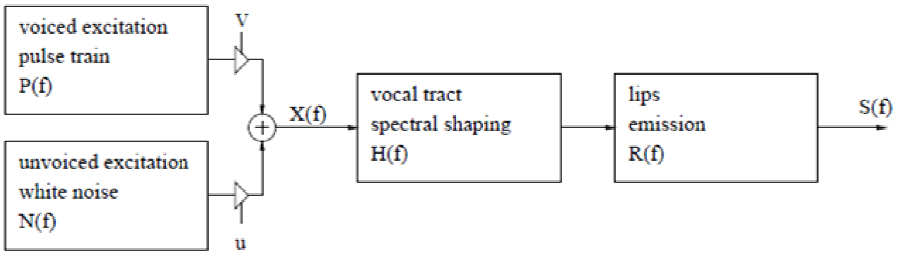


Figure 2. Simple model of speech production.

To analyze the speech signal, it is necessary to know that the direct computation of the power spectrum from the speech signal results in a spectrum containing “ripples” caused by the excitation spectrum X(f). “Ripple” refers to periodic variation in insertion loss with frequency of a filter. Not all filters contain ripples; some, such as the Butterworth filter, monotonically increase insertion loss with frequency. The ripple is not usually strictly linearly or periodic, as can be seen in the example plot in Figure 3. *[2]*

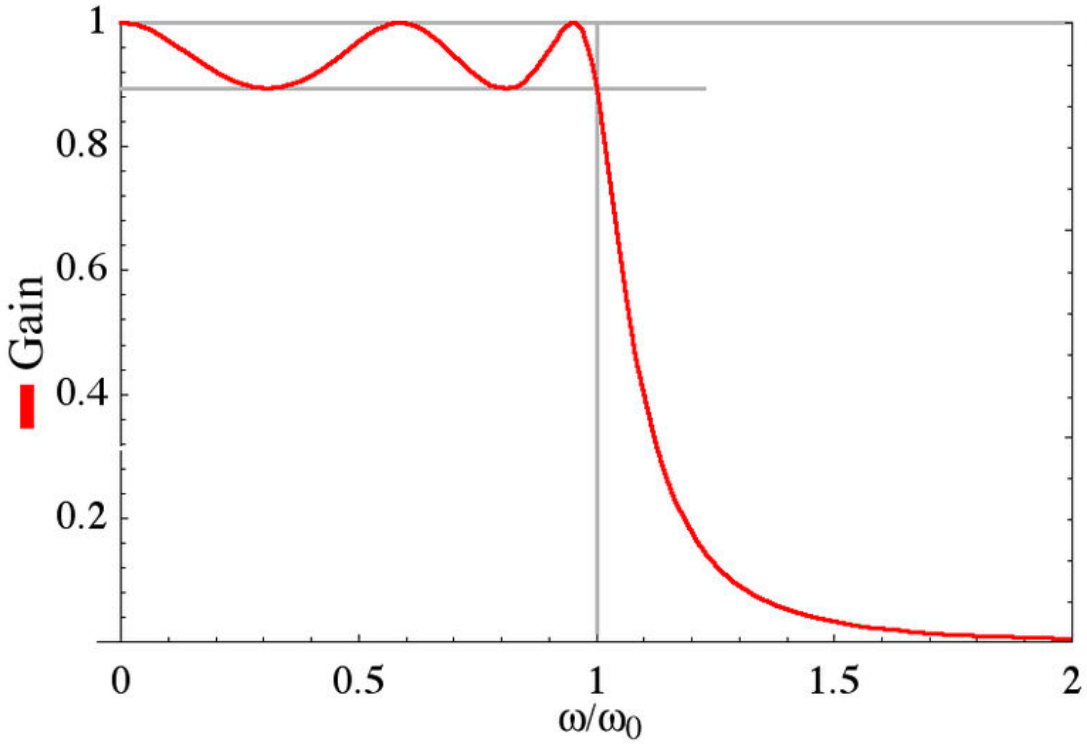


Figure 3. Ripple on a Chebyshev filter.

Depending on the implementation of the acoustic preprocessing, however, special transformations are used to separate the excitation spectrum X(f) from the spectral shaping of the vocal tract H(f). Thus, a smooth spectral shape (without the ripples), which represents H(f), can be estimated from the speech signal. Most speech recognition systems use the so-called mel

frequency cepstral coefficient (MFCC) and its first (and sometimes second) derivative in time to better reflect dynamic changes. *[3]*

The following sections will explain what are and its influence in this project for the MFCC.

1.1.3. Characteristics of Speech Signal

Any signal can be characterized as follows:

- Speech signal is quasi-stationary.
- The bandwidth of any signal is 4 kHz.
- Fundamental frequency is between 80 and 350 Hz.
- It is periodic for voiced signal.
- There are peaks in the spectral distribution of energy at $(2n-1)*500\ Hz; n = 1,2,3$ ...
- The envelope of the power spectrum of the signal decreases with increasing frequency (-6dB per octave).

These characteristics are generally present in any audio signal, but where those features come from varies. For instance, audio signals have a bandwidth much greater than 4kHz. However, for analog phones, a bandwidth of 4 kHz for the speech signal is sufficient for understanding the human voice. Many of the applications for speech signal processing are related to phone usage, so we will take this bandwidth as the baseline assumption.

1.2. Basics of Digital Signal Speech Processing

The fundamental purpose of speech is communication, i.e., the transmission of messages. According to Shannon's information theory, a message represented as a sequence of discrete symbols can be quantified by its information content in bits, and the rate of transmission of information is measured in bits/second (bps). That transformation into a sequence of known symbols allows us to treat the signal and process it.

In speech production, as well as in many human-engineered electronic communication systems, the information to be transmitted is encoded in the form of a continuously varying (analog) waveform that can be transmitted, recorded, manipulated, and finally decoded by a human listener. In the case of speech, the fundamental analog form of the message is an acoustic waveform, the speech signal.

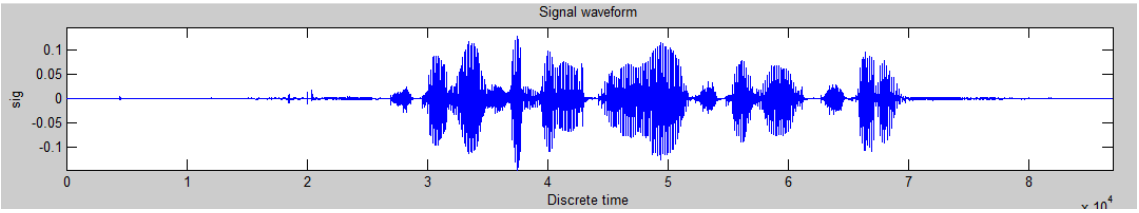


Figure 4. A speech waveform.

1.2.1. Basic Characteristics of Speech Processing: Frame Energy

A fundamental principle of speech processing is the calculation of the energy of a signal; in terms of this project specifically, we must calculate the energy of a frame. The sign is to be broken down into frames of 32 ms, and each frame is processed each time. Each frame is comprised of 256 samples with a 50% overlapping of the foregoing. These values have been selected after several tests that determined that those specific values obtained optimized results. The results of those tests and the choice of selected values will be examined in later chapters.

The energy will be calculated thus:

Let $\mathrm{x(i)}$ be the $\mathrm{i^{th}}$ sample of speech. If the length of the frame were k samples, then the $\mathrm{j^{th}}$ frame can be represented in a time domain by a sequence as

$$\mathrm{f_j} = \{\mathrm{x(i)}\}_{\mathrm{i=(j-1)k+1}}^{\mathrm{jk}}$$

- $0 \leq \mathrm{K} \geq 256\ \mathrm{samples}$

We associate energy $\mathrm{E_j}$ with the $\mathrm{j^{th}}$ frame as:

$$\mathrm{E_j} = \frac{1}{\mathrm{k}} \sum_{\mathrm{i=(j-1)k+1}}^{\mathrm{jk}} \mathrm{x^2(i)}$$

where:

- E_j is the energy of the j^{th} frame

1.2.2. Linear Predictive Coding (LPC) Model

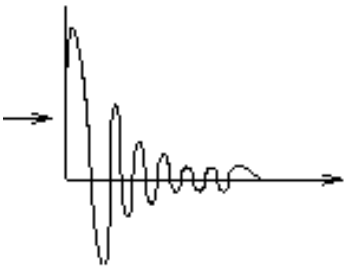
Linear predictive coding (LPC) is a digital method for encoding an analogue signal in which a particular value is predicted by a linear function of the past values of the signal. Human speech is produced in the vocal tract, which can be approximated as a variable diameter tube. The linear predictive coding (LPC) model is based on a mathematical approximation of the vocal tract represented by this tube of varying diameter. At a particular time, *t*, the speech sample *s(t)* is represented as a linear sum of the previous samples. The most important aspect of LPC is the linear predictive filter, which allows the value of the next sample to be determined by a linear combination of previous samples. Since there is information loss in linear predictive coding, it is a lousy way of compression.

1.2.2.1. Linear Prediction Model

Linear prediction is a good tool for analysis of speech signals. Linear prediction models the human vocal tract as an infinite impulse response (IIR) system that produces the speech signal. For vowel sounds and other voiced regions of speech, that have a resonant structure and high degree of similarity over time shifts that are multiples of their pitch period, this modeling produces

an efficient representation of the sound. Figure 5 shows how the resonant structure of a vowel could be captured by an IIR system. *[4] [5]*

$$y[n] = \frac{1}{a_0}\,(b_0x[n] + b_1x[n-1] + \ \ldots + \ b_px[n-p] - a_1y[n-1] - \ a_2y[n-2] \\ - \ \ldots - \ a_qy[n-q]$$



where:

- *p is the feedforward filter order*
- b_i *are the feedforward filter coefficients*
- *q is the feedback filter order*
- a_i *are the feedback filter coefficients*
- $x[n]$ *is the input signal*
- $y[n]$ *is the output signal*

Figure 5. Linear Prediction (IIR) model of speech.

The linear prediction problem finds the coefficients $\mathrm{a_k}$ which results in the best prediction (by minimizing mean-squared prediction error) of the speech sample ***s[n]*** in terms of the past samples $s[n-k], k = \{1, \ldots, P\}$. The predicted sample $\widehat{s[n]}$ is then given by *Rabiner and Juang:*

$$\hat{s}[n] = \sum_{k=1}^{P} a_k\, s[n-k]$$

where P is the number of past samples of ***s[n]*** that we wish to examine.

Next we derive the frequency response of the system in terms of the prediction coefficients a_k. In the previous equation, when the predicted sample equals the actual signal we have:

$$\hat{s}[n] = \sum_{k=1}^{P} a_k\, s[n-k]$$

$$s(z) = \sum_{k=1}^{P} a_k\, s(z)z^{-k}$$

$$s(z) = \frac{1}{1 - \sum_{k=1}^{P} a_k\, z^{-k}}$$

The optimal solution to this is *Rabiner and Juang [8]*:

$$a = (a_1 a_2 \dots a_P)$$

$$a = R^{-1} r$$

Due to the *Toeplitz* property of the *R* matrix (it is symmetrical with equal diagonal elements), an efficient algorithm is available for computing a without the computational expense of finding R^{-1}. The Levinson-Durbin algorithm is an iterative method of computing the predictor coefficients a *Rabiner and Juang [8]*.

Initial step: $E_0 = r_{ss}[0], i = 1$ for *i=1... P*

Steps:

- $k_i = \frac{1}{E_{i-1}}(r_{ss}[i] - \sum_{j=1}^{i-1}(\alpha_{j,i-1} r_{ss}[|i-j|]))$
- $\alpha_{j,i} = \alpha_{j,i-1} - k_i \alpha_{i-j,i-1}$; *j=[1,..., k-1]*
- $\alpha_{i,i} = k_i$
- $E_i = (1 - {k_i}^2) E_{i-1}$

$$r = (r_{ss}[1]\ r_{ss}[2] \dots r_{ss}[P])^T$$

$$R = \begin{pmatrix} r_{ss}[0] & r_{ss}[1] & \cdots & r_{ss}[P-1] \\ r_{ss}[1] & r_{ss}[0] & \dots & r_{ss}[P-2] \\ \vdots & \dots & \ddots & \vdots \\ r_{ss}[P-1] & r_{ss}[P-2] & \cdots & r_{ss}[0] \end{pmatrix}$$

To reduce the discontinuity between segments, we do not clear the states of the IIR model from one segment to the next. Instead, we load the new set of reflection coefficients, $\mathrm{k_i}$, and continue with the lattice filter computation.

Understanding this model will make it possible to understand the workings of the autoregressive model (AR) that will be used to calculate the coefficients. The LPC is arguably a precursor to these coefficients.

1.2.3. Autoregressive Model (AR)

The AR model plays an important role in the development of the project. Which is used for computation of the cepstral coefficients from the AR coefficients for the implementation of VAD will be determined later.

An autoregressive model depends on a limited number of parameters that are estimated from measured noise data. Several methods exist for estimating the autoregressive parameters, such as least squares, Yule-Walker, and Burg's method. For large data samples, these estimation techniques should lead to approximately the same parameter estimates. *[13]*

1.2.3.1 Theory of Autoregressive Model

The successive samples y_t of an autoregressive process linearly depend on their predecessors:

$$y_t + a_1 y_{t-1} + a_2 y_{t-2} + \cdots + a_p y_{t-p} = \eta_t$$

Where:

- a_i: autoregressive coefficients
- η_t: stationary purely random process with zero mean

The autocovariance function R_t for delays from 0 to *p* is related to the autoregressive coefficients a_i through the Yule-Walker equation for the autoregressive process:

$$\begin{pmatrix} R_0 & \cdots & R_{p-1} \\ \vdots & \ddots & \vdots \\ R_{p-1} & \cdots & R_0 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{pmatrix} = - \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_p \end{pmatrix}$$

An estimated autoregressive model of the same order *p* can be written as:

$$y_t + \widehat{a_1} y_{t-1} + \widehat{a_2} y_{t-2} + \cdots + \widehat{a_p} y_{t-p} = \widehat{\eta_t}$$

Where:

- $\widehat{a_i}$: autoregressive coefficients estimated
- $\widehat{\eta_t}$: estimated innovations

The difference between the autoregressive process and corresponding autoregressive model (first and third equation) should be taken into account.[5] By using the last one, each data sample can be predicted from its predecessors:

$$\widehat{y_t} = -\sum_{i=1}^{p} \widehat{a_i}\, y_{t-i}$$

Since the samples y_t cannot be predicted exactly, a residue is introduced, which is defined as the difference between the measured value and the estimated value:

$$\text{residue} = y_t - \widehat{y_t} = \widehat{\eta_t}$$

which means that the residue is equal to the estimated innovations.

It is assumed in these equations that the autoregressive model order p is known. In practice, the model order has to be estimated as well, which is usually done using Akaike's criterion. *[15]*

Suppose that the estimation realization y consists of N data points (an estimation realization contains those data points that are used for parameter estimation). Three methods of autoregressive-coefficients estimation from these data samples shall be considered here: the least-squares approach (LS), the Yule-Walker approach (YW), and Burg's method (Burg).

1.2.3.2 Autoregressive coefficients calculation

- **LS**: the total squared residue over the data samples p+1 to N is minimized, leading to a system of linear equations:

$$\begin{pmatrix} c_{11} & \cdots & c_{1p} \\ \vdots & \ddots & \vdots \\ c_{p1} & \cdots & c_{pp} \end{pmatrix} \begin{pmatrix} \widehat{a_1} \\ \widehat{a_2} \\ \vdots \\ \widehat{a_p} \end{pmatrix} = - \begin{pmatrix} c_{01} \\ c_{02} \\ \vdots \\ c_{0p} \end{pmatrix}$$

  in which the matrix elements

$$c_{ij} = \frac{1}{N-p} \sum_{i=p+1}^{N} y_{t-i}\, y_{t-j}$$

  form and unbiased estimate of the autocovariance function for delay *i-j*.

- **YW**: The first and last p data points are also included in the summation of c_{ij} resulting in:

$$\begin{pmatrix} \widehat{R_0} & \cdots & \widehat{R_{p-1}} \\ \vdots & \ddots & \vdots \\ \widehat{R_{p-1}} & \cdots & \widehat{R_0} \end{pmatrix} \begin{pmatrix} \widehat{a_1} \\ \widehat{a_2} \\ \vdots \\ \widehat{a_p} \end{pmatrix} = - \begin{pmatrix} \widehat{R_1} \\ \widehat{R_2} \\ \vdots \\ \widehat{R_p} \end{pmatrix}$$

  in which the matrix elements $\widehat{R_t}$ constitute a biased estimate of the autocovariance function. *[16]*

  The Levinson-Durbin algorithm provides a fast solution to a system of linear equations containing a Toeplitz-style matrix such as YW.

- **Burg**: This method is currently regarded as the most appropriate. Unlike the other methods, which estimate the autoregressive coefficients directly, Burg's method first estimates the reflection coefficients, which are defined as the last autoregressive coefficient estimate for each model order p. From these, the parameter estimates are determined using the Levinson-Durbin algorithm. The reflection coefficients constitute unbiased estimates of the partial correlation coefficients.

Usually, these estimation methods lead to approximately the same results for the autoregressive parameters. Once these have been estimated from the time series y, the autoregressive model can be applied to an independent prediction realization x of the same stochastic process. In terms of x, the autoregressive process can be written as:

$$x_t + a_1 x_{t-1} + a_2 x_{t-2} + \cdots + a_p x_{t-p} = \varepsilon_t$$

in which the innovation process ε_t is statistically identical to the innovation process η_t . The corresponding autoregressive model can be written as:

$$x_t + \widehat{a_1} x_{t-1} + \widehat{a_2} x_{t-2} + \cdots + \widehat{a_p} x_{t-p} = \widehat{\varepsilon_t}$$

in which $\widehat{a_i}$ are the autoregressive coefficients estimated from realization y and $\widehat{\varepsilon_t}$ are the estimated innovations. Each data sample can be estimated from its predecessors:

$$\widehat{x_t} = -\sum_{i=1}^{p} \widehat{a_i}\, x_{t-i}$$

The difference between the measured value and the estimated value is now defined as the prediction error:

$$\text{residue} = x_t - \widehat{x_t} = \widehat{\varepsilon_t}$$

The prediction error is therefore equal to the estimated innovation. Each prediction error can be calculated once the actual value of the data point is measured.

A clear distinction should be made between the residue and the prediction error and their variances. [*14]* The residual variance *var($\widehat{\eta_t}$)t* is a measure for the fit of the autoregressive model to data that have been used for estimating the autoregressive parameters, and can be estimated from the realization y, which is used for the parameter estimation:

$$\widehat{\text{var}}(\widehat{\eta_t}) = \frac{1}{N-p}\sum_{t=p+1}^{N}(y_t - \widehat{y_t})^2$$

For the prediction of future data, instead of the residual variance, the variance of the prediction error var($\widehat{\varepsilon_t}$) is essential. If the independent prediction realization x contains N' data samples, the prediction error variance can be estimated from the sample variance:

$$\widehat{\text{var}}(\widehat{\varepsilon_t}) = \frac{1}{N'-p}\sum_{t=p+1}^{N'}(x_t - \widehat{x_t})^2$$

The LS parameter estimation is based on the minimization of the residual variance. However, such a minimization does not imply that the variance of the prediction error is minimized as well. Since the minimization of the prediction error variance is usually our goal, the LS estimation of the autoregressive parameters is not necessarily superior to YW or Burg's method.

Burg's algorithm is really related to the algorithm designed for the project as the main tool for computation of the AR model. Through his method, we find one of the input parameters for the computation of the cepstral coefficients from the AR coefficients. The main reasons to select the Burg algorithm are because it is robust and has an efficient computation, so it gives our VAD algorithm robustness and efficiency.

This section clarifies theoretical aspects of Burg's algorithm to further elucidate his contribution to the project.

1.2.3 Cepstral Analysis

Cepstral analysis is a key concept in the development of the algorithm. This analysis will lead to the cepstral distance that is the basis of the algorithm for

Voice Activity Detection. This section will describe fully all these concepts and how to calculate them.

A signal comes out of a system due to the input excitation and also the response of the system. From the signal processing point of view, the output of a system can be treated as the convolution of the input excitation with the system response. At times, it is necessary to identify each of the components separately for study and/or processing. The process of separating the two components is called deconvolution.

In the first case, if we know the input excitation, then the system component can be separated /constructed by exciting the system with the inputs and collecting its responses. This is what is done in same channel estimation problems. In the second case, if we know the system response, then the input excitation can be recovered using the inverse filter theory concept, For instance, via the Linear Prediction (LP) analysis of speech to recover excitation. There is yet another type of deconvolution in which the assumption is that both input excitations and system responses are unknown.

Speech is composed of an excitation source and vocal tract system components. To analyze and model the excitation and system components of the speech independently and then use that result in various speech processing applications, these two components have to be separated from speech. The objective of cepstral analysis is to separate speech into its source and system components without any a priori knowledge about source and/or system.

According to the source filter theory of speech production, voiced sounds are produced by exciting the time-varying system characteristics with periodic impulse sequence. Unvoiced sounds are produced by exciting the time-varying system with a random noise sequence. The resulting speech can be considered the convolution of the respective excitation sequence and vocal tract filter characteristics. If e(n) is the excitation sequence and h(n) is the vocal tract filter sequence, then the speech sequence s(n) can be expressed as:

$$s(n) = e(n) * h(n)$$

This can be represented in frequency domain as:

$$S(\omega) = E(\omega) \cdot H(\omega)$$

The second equation indicates the multiplication of excitation and system components in the frequency domain for the convolved sequence of the same time domain. The speech sequence has to be deconvolved into the excitation and vocal tract components in the time domain. For this, multiplication of the two components in the frequency domain has to be converted to a linear combination of the two components. Cepstral analysis is used to transform the multiplied source and system components in the frequency domain to linear combination of the two components in the cepstral domain. *[5][6][7]*

1.2.3.1 Basics Principles of Cepstral Analysis

From the last equation, the magnitude spectrum of given speech sequence can be represented as:

$$|S(\omega)| = |E(\omega)| \cdot |H(\omega)|$$

To linearly combine the E(ω) and H(ω) in the frequency domain, logarithmic representation is used. The logarithmic representation of last equation will be:

$$\log|S(w)| = \log|E(w)| + \log|H(w)|$$

As indicated in the previous equation, the *log* operation transforms the magnitude speech spectrum in which the excitation component and vocal tract component are multiplied into a linear combination of these components, i.e., log operation converted the $*$ operation into $\cdot$ operation in the frequency domain. The separation can be done by taking the Inverse Discrete Fourier Transform (IDFT) of the linearly combined log spectra of excitation and vocal tract system components. It should be noted that IDFT of linear spectra transforms back to the time domain, but the IDFT of log spectra transforms to frequency domain or the cepstral domain, which is similar to the time domain. In the frequency domain, the vocal tract components are represented by the slowly varying components concentrated near the lower frequency region and excitation components are represented by the fast varying components at the higher frequency region.

1.2.3.1.1 Linear prediction cepstral coefficients: LPC cepstrum

There are other options for calculating cepstrum parameters besides IDFT. It is possible to make the calculations based on LPC model. Linear Prediction Cepstrum Coefficients (LPC) are represented in the cepstrum domain. The idea of LPC is based on the speech production model in which the characteristics of the vocal tract can be modeled by an all-pole filter.

LPC is simply the coefficients of this all-pole filter and is equivalent to the smoothed envelope of the log spectrum of the speech. LPC can be calculated either by the autocorrelation or covariance methods directly from the windowed portion of speech. The LPCC [18][19] were acquired from the LPC as:

$$\mathrm{LPCC} = \mathrm{LPC_t} + \sum_{k=1}^{i-1} \frac{k-i}{i} \mathrm{LPCC_{i-k}} \mathrm{LPC_k}$$

LPCC have been widely used for a few decades and have been proven to be more robust and reliable than LPC. However, LPCC also inherited the disadvantages of LPC. One of the main disadvantages is that LPC approximates speech linearly at all frequencies. This is inconsistent with the perception of human hearing. Also, LPC includes the details of the high frequency portion of a speech that contains mostly noise. This inclusion of noise information may affect the system's performance.

1.2.3.1.2 Mel Frequency Cepstral Coefficients (MFCC): mel-cepstrum

In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transformation of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) collectively make up an MFC.

The motivation for using Mel-Frequency Cepstrum Coefficients is the fact that the auditory response of the human ear resolves frequencies non-linearly. The mapping from linear frequency to mel-frequency is defined as:

$$\mathrm{f_{mel}} = 2595 * \log_{10}(1 + \frac{\mathrm{f}}{700})$$

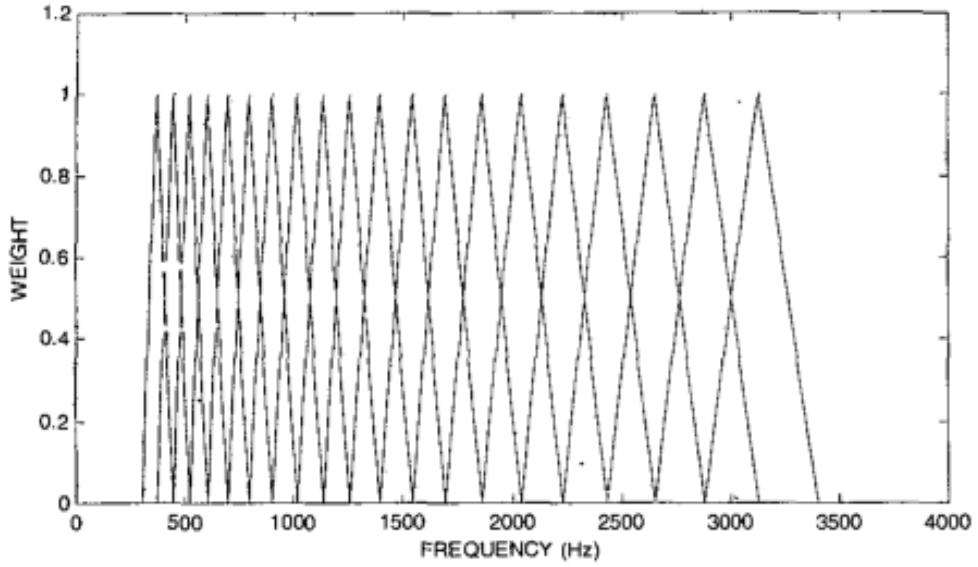


Figure 6. Filters for generating MFCCs with band-limiting between 300 to 3400Hz.

The MFCC were computed using the Discrete Cosine Transform:

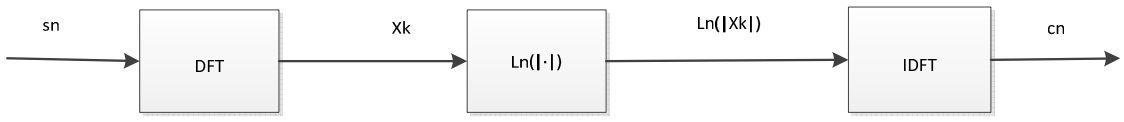
$$MFCC_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}(j - 0.5)\right)$$

where *N* is the number of bandpass filters and m_j is the log bandpass filter output amplitudes.

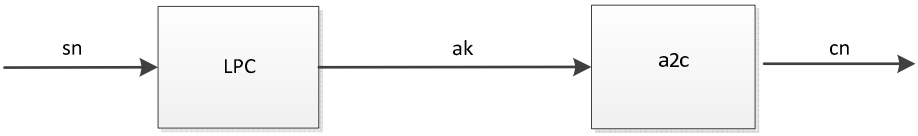
It has the benefit that it is capable of capturing the phonetically important characteristics of speech. Also band-limiting can easily be employed to make it suitable for telephone applications. A small drawback is that MFCCs are more computationally expensive than LPCC due to the Fast Fourier Transform (FFT) at the early stages to convert speech from the time to the frequency domain. *[19][20]*

1.2.3.1.3 Computation Cepstrum

- Computation of DFT based on cepstral coefficients

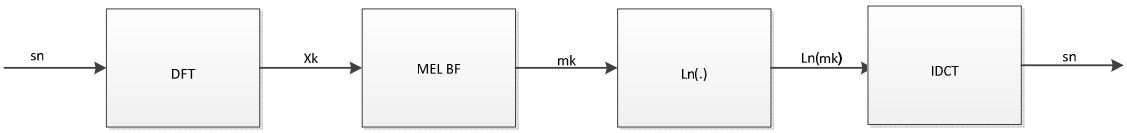


- Computation of LPC cepstral coefficients



This final option is the selected for the computation of AR coefficients in the project:

- Computation of MEL cepstral coefficients



1.2.3.2 Cepstral Analysis Applications

The next two concepts are the fundamental principles of the implementation of the algorithm. The cepstral distance is what determines whether the frame is active or inactive. Cepstral distance is calculated and entered a previous frame and greater distance between them, implying a greater difference and therefore the voice activity detection. The way to calculate it is:

1.2.3.2.1 Spectral Distance

This is a distance measurement (expressed in dB) between two different spectra *[8]*.

$$L_2 = \int_{-\pi}^{\pi} ln \frac{|S_1(e^{j\theta}|^2}{|S_2(e^{j\theta}|^2} d\theta$$

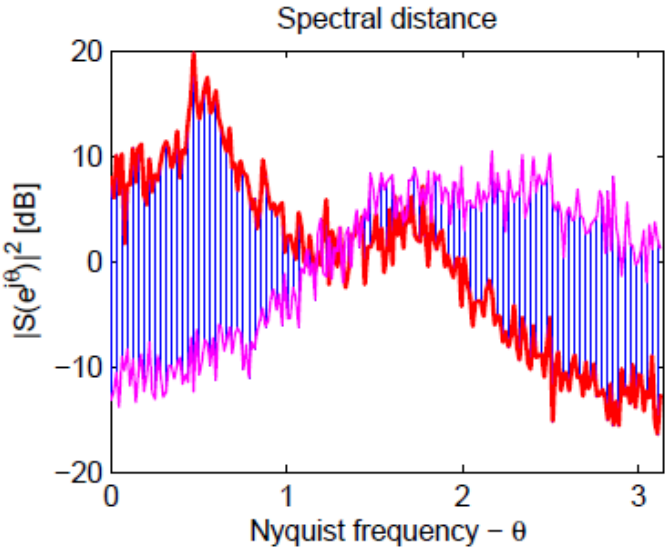


Figure 7. Spectral distance

1.2.3.2.2 Cepstral Distance

$$CD = \sqrt{(c_s[0] - c_x[0])^2 + 2\sum_{k=1}^{L}(c_s[k] - c_x[k])^2}$$

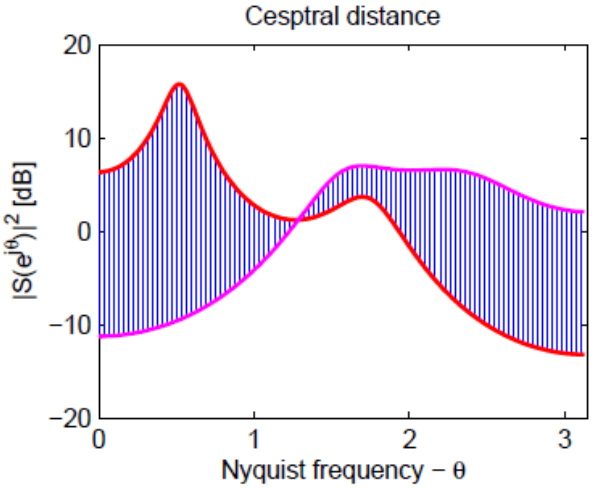


Figure 8. Cepstral distance

1.3 Speech Recognition

A speech recognition system, at its most elementary level, is comprised of a collection of algorithms drawn from a wide variety of disciplines, including statistical pattern recognition, communication theory, signal processing, combinatorial mathematics, and linguistics, among others. Although each of these areas relies on different recognizers to varying degrees, perhaps the greatest common denominator of all recognition systems is the signal-processing front end, which converts the speech waveform into some type of parametric representation (generally at a considerably lower information rate) for further analysis and processing. *[4]*

A wide range of possibilities exists for parametrically representing the speech signal, including the short time energy, zero crossing rates, level crossing rates, and other related parameters. Probably the most important parametric representation of speech is the short time spectral envelope. Spectral analysis is therefore generally considered the core of the signal-processing front end in a speech-recognition system. Two of the most dominant methods of spectral speech recognition system are the filter-bank spectrum analysis model and the linear predictive coding (LPC) spectral analysis model, which will be used to implement the algorithm.

As previously mentioned, a digit recognizer has been added to give more value to the project. This is Dynamic Time Warping (DTW) algorithm is a time series alignment algorithm developed originally for speech recognition. It aims at aligning two sequences of feature vectors by warping the time axis iteratively until an optimal match (according to a suitable metric) between the two sequences is found.

The algorithm implemented for the final application is one of the simplest DTW algorithms. It is mainly based on the calculation of mel-frequency cepstrum computation for each of the digits of the pattern available previously

recorded and the cepstrum computation of the input signal. The next step is to calculate the distance between the cepstrum computation of each digit and the input signal. These distances are evaluated, and the smallest distance corresponds to the digit that contains the signal.

CHAPTER 2. VOICE ACTIVITY DETECTION

This chapter describes the basics for implementation to the final algorithm developed for the project. The algorithm is based on Voice Activity Detection and Start-End-Point detection algorithms. Finally, the peculiarities of the algorithm designed for this project and how it works will be detailed.

2.1. Designing Algorithms

An important point of signal processing that is relevant for the project is voice activity detection. A Voice Activity Detection algorithm that can detect active and inactive frames has been used for this purpose. This section will describe the basic principles for a general design; details of the algorithm will be described in the next chapter.

2.1.1. Basics of VAD Algorithm

The process of separating conversational speech and silence is called voice activity detection (VAD). It was first investigated for use on Time Assigned Speech Interpolation (TASI) systems. VAD is an important enabling technology for a variety of speech-based applications, including speech recognition, speech encoding, and hands-free telephony.

The variety and varying nature of speech and background noise make it challenging. Earlier algorithms for VAD are based on the Itakura LPC distance measure, energy levels, timing, pitch, zero crossing rates, cepstral features, adaptive noise modeling of voice signals, and the periodicity measure. Unfortunately, these algorithms have some problems for low SNR values, especially when the noise is non-stationary. An acceptable accuracy cannot be achieved since most algorithms rely on a threshold level by comparison. This threshold level is often assumed to be fixed or calculated in the silence (voice-inactive) intervals.

Differentiation of the voiced signal into speech and silence is done on the basis of speech characteristics. The signal is sliced into contiguous frames. A real-valued non-negative parameter, the average energy content, is associated with each frame. If this parameter exceeds a certain threshold, the signal frame is classified as ACTIVE; else it is INACTIVG. We also refer to these INACTIVE frames as noise frames. *[10][11]*

2.1.2. VAD Based on Adaptive Threshold

2.1.2.1. Choice of Frame Duration

To develop a better method to process the signal, we divide it into frames of equal size. The specifications for our detection speech algorithm are:

- 16 kHz sampling frequency single channel (mono) recording;
- 256 levels of linear quantization (8 Bit PCM);
- single channel (mono) recording;

- frame duration of 32 ms. An audio signal is constantly changing, so to simplify processing, we assume that on short time scales, the audio signal does not change much (meaning statistically, i.e., statistically stationary, although obviously the samples are constantly changing on even short time scales). This is we have chosen to frame the signal into 32ms frames. If the frame is much shorter, we do not have enough samples to get a reliable spectral estimate; if it is longer, the signal changes too much throughout the frame;
- 16.000 samples/second*32ms=512 samples/frame.

2.1.2.2. Initial Value of Threshold

Obtaining the threshold is considered the first two inactive frames that are referenced in the background noise level. The initial estimate of energy is obtained by taking the mean of the energies of each frame as: *[21]*

$$\mathrm{E_r} = \frac{1}{\mathrm{v}} \sum_{\mathrm{m}=0}^{\mathrm{v}} \mathrm{E_m}$$

where:

- E_r is initial threshold estimate; and
- v is the number of model background noise frames: 2.

2.1.2.3. ACTIVE or INACTIVE

The energy of a frame is a reasonable parameter for classifying frames as ACTIVE or INACTIVE. The energy of ACTIVE frames is higher than that of INACTIVE frames. The classification rule is:

$$\mathrm{IF\ E_j} > k\mathrm{E_r}\ \text{where}\ \mathrm{k} > 1 \rightarrow Frame\ is\ ACTIVE$$
$$\mathrm{ELSE} \qquad \text{Frame is INACTIVE}$$

In this equation, $\mathrm{E_r}$, represents the energy of noise frames, while $\mathrm{kE_r}$, is the threshold used in the decision-making.

Because it has a scaling factor, k allows a safe band for the adaptation of $\mathrm{E_r}$, and hence, the threshold.

2.1.2.4. Adaptive Threshold

Since background disturbance is non-stationary, an adaptive threshold is more appropriate. The rule to update the threshold value can be found in:

$$E_{rnew} = (1-p)E_{rold} + pE_{rsilence}$$

where,

- E_{rnew} is the updated value of the threshold
- E_{rold} is the previous energy threshold
- $E_{rsilence}$ is the energy of the most recent noise frame
- $0 < p < 1$ parameter *p* is chosen to consider the impulse response of previous equation as a first order filter.

2.1.2.5. VAD Based on Cepstral Analysis

VAD can be based on applying the cepstral analysis to detection of voice. To do this, the cepstral distance is calculated between the current and background frames. The detection method is based on having greater distance between the two frames, meaning there is a greater difference between the two and therefore there is voice activity.

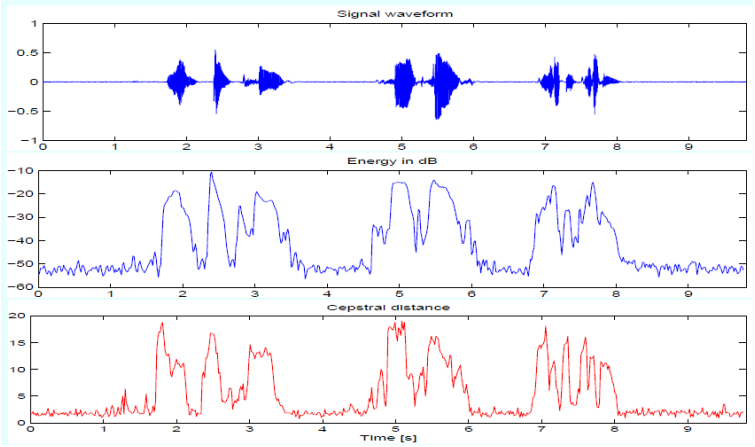


Figure 9. Cepstral distance in VAD.

2.1.3. Start- and End-Point Detection

Once it has been implemented, the VAD separates the signal into parts in which the voice is detected separately from the silences. The next step is to identify the beginning and end of the signal. A criterion should be implemented to determine the start-of-speech (SOS) and end-of-speech (EOS) points.

2.2. Algorithm Implemented for the Project

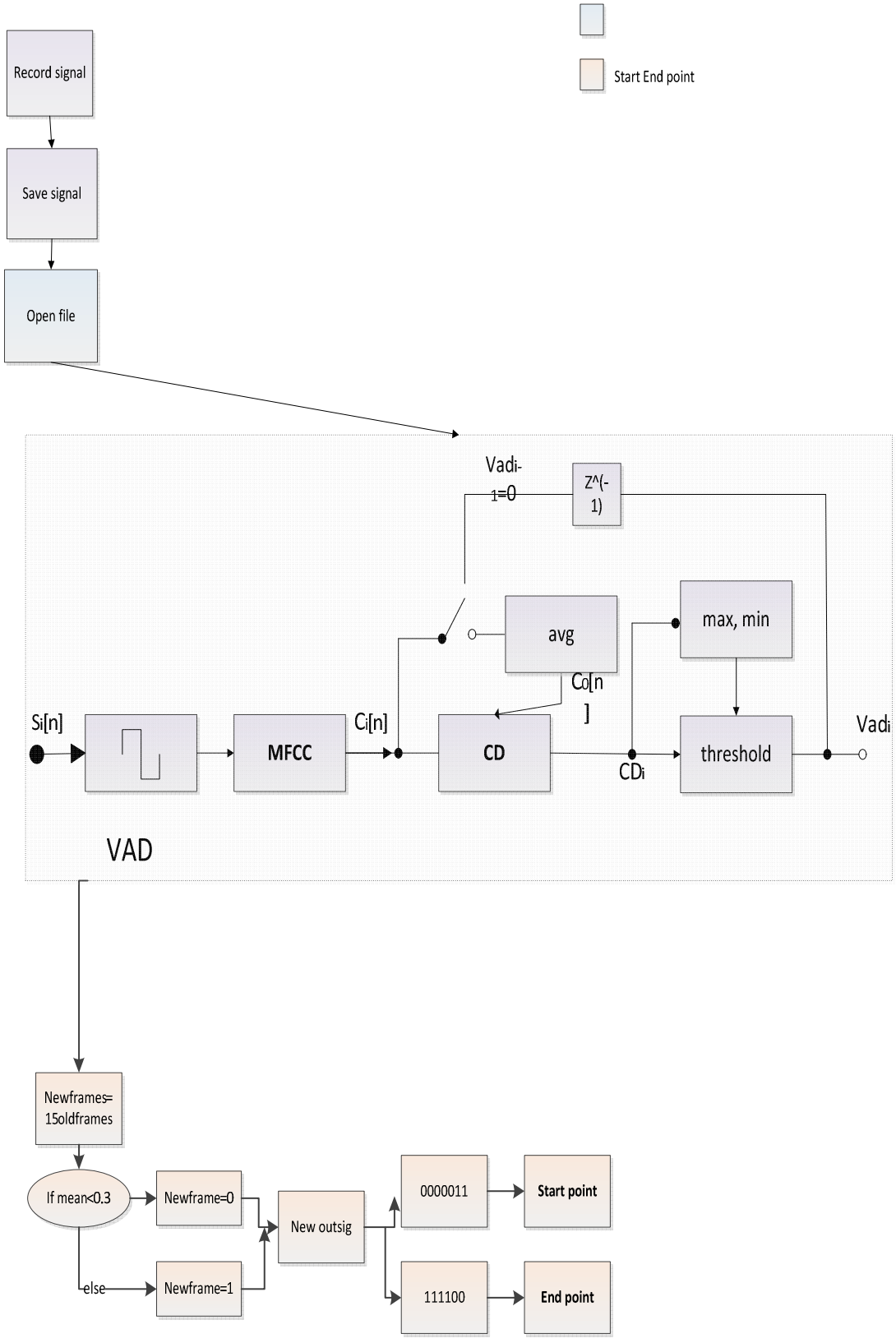


Figure 10. Designed algorithm.

Figure 10 shows the operation of the final algorithm that has been used to implement the application for the voice activity detection.

2.2.1. Reading and Saving Process

As shown in Figure 10, the algorithm can be divided into two phases. The first is concerned with the Voice Activity Detection algorithm depicted in blue. The second reflects the start-and-end point detection.

The first step is to record a signal during a given time. In this case, we decided to set this time in 8 seconds, because the final application will be a voice recognizer based on patterns that contain digits 0 to 9. Therefore, 8 seconds of time is sufficient for subsequent signal processing. The measured signal is stored in a file that is open later for processing. The reason for saving the signal is that it should be continuously stored in a buffer and therefore could be processed continuously and indefinitely. For this occasion, a definite time of 8 seconds was chosen.

2.2.2. VAD Algorithm

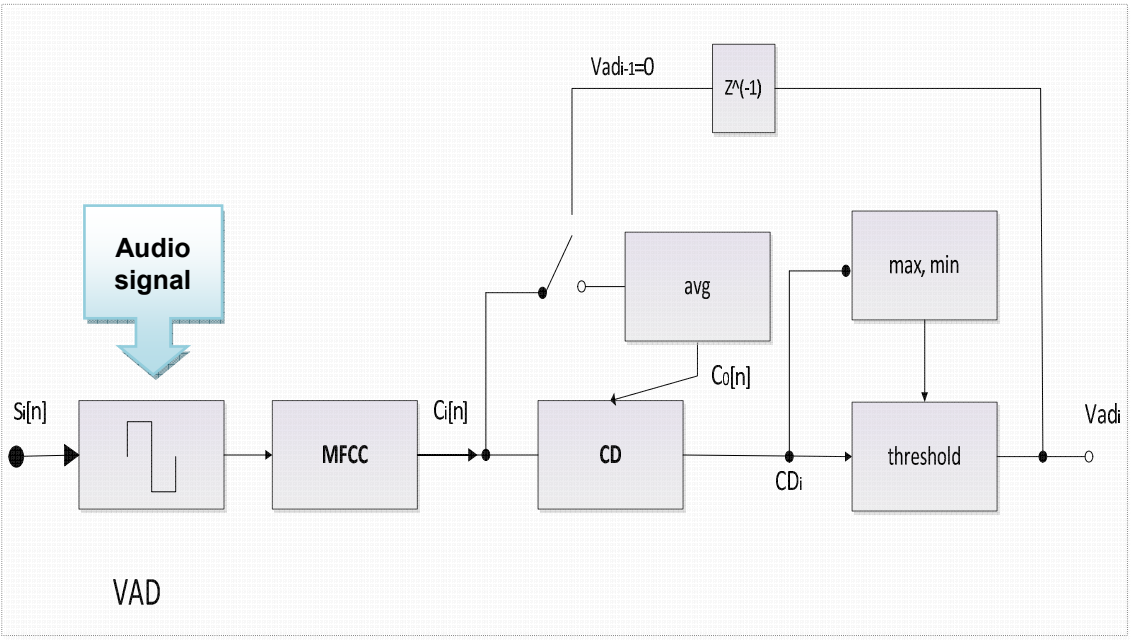


Figure 11. VAD Algorithm designed for the project.

Once the signal is in a file, it is going to read slowly with each length of time equaling *wlen/2*, where *wlen* = 256 points. The first two readings obtain frame 1 and frame 2, which will be used to obtain the reference values with which to compare the following frames. In this step, we used the Burg model for the computation of AR model and A2C function for computation of cepstral coefficients from AR coefficients. We also calculated the maximum (Dmax) and minimum (Dmin) distance and the difference (Dyn) between them. We will define forgetting parameters of maximum and minimum distance (case of increase and decrease) (qmax1, qmax2, qmin1, qmin2).

After calculating these parameters, we will choose the cycle for the algorithm to calculate the entire signal. A while loop with the new data of each

read (*wlen*/2) are new data and previous samples old data, will be used. Each frame will consist of old data + new data (i.e., by 256 points). Then, we will calculate *Dmax*, *Dmin* and *Dyn* for each, which determine whether the frame being evaluated is the voice activity (if D> *Dyn & Dp> 0.25*) or not (otherwise).

2.2.3. Start and End Point Detection

At the end of the foregoing process, *outsig* signal is obtained, which is a signal formed by a sequence of 0 and 1 with frames of 32 ms.

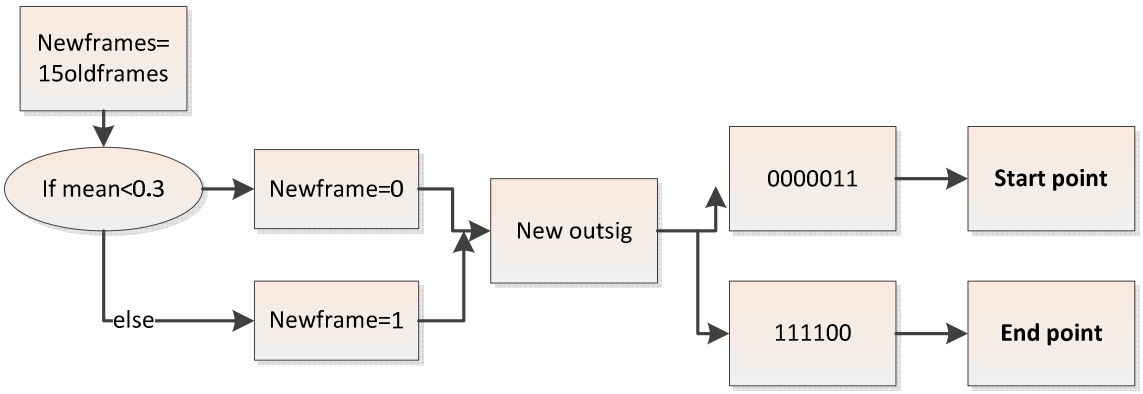


Figure 12. Start-and-End point detection algorithm implemented.

The next step is to build frames of 0.25 seconds, so 15 frames are a new frame. At this point, start-end point detection begins.

We set a decision threshold of 30%. If the mean of 1 to 15 frames is less than 0.3 frame, we identify it as an inactive frame; otherwise voice is detected, so it will designated a active frame. The next step is to check whether the sequence is 0111, because this is considered the beginning of the signal. The sequence 11100 determines the end. As a final step, we determine that the results are optimized if the marked departure of the signal after subtracting 1000 points and at the end point has 2500 points.

Once it is obtained, only the voiced signal is clipped and saved in another file.

CHAPTER 3. IMPLEMENTATION

The third chapter will describe the process of the application deployment. First, some important implementation issues in MATLAB and will be described and the tools that are necessary for this project will be identified. Second, we will explain the process for online processing. Third, we will analyze real-time speech processing. Finally, we will describe my application.

3.1. General Issues of MATLAB Implementation

The application has been fully developed in MATLAB, as were the algorithm and the application.

To carry out the project, we had to install of the MATLAB program (version R2011b, compatible with Windows operating system of 32 bits). We also installed the signal processing toolbox, since we will need certain functions it contains for further signal processing.

In additions, some functions have been necessary for the development of the algorithm. The described functions had been previously developed to the beginning of the project and provided to me by my tutor for my thesis. These are useful for developing the VAD algorithm that can detect the voice activity and silence periods.

- Function A2C: is included for computation of cepstral coefficients from AR coefficients.
  - $c = a2c(a, p, c_p)$
    - *a*: vector of AR coefficients (without *a[0] = 1*)
    - *p*: order of AR model (number of coefficients without *a[0]*)
    - *c*: vector of cepstral coefficients (without *c[0]*)
    - c_p: order of cepstral model (number of coefficients without *c[0]*)
- Function BURG: is used for computation of AR model using Burg's algorithm.
  - $[a, \alpha, r_c] = burg(x, p)$
  - Input parameters:
    - *x*: processing frame
    - *p*: order of LPC model
  - Output parameters:
    - *a:* autoregressive coefficients
    - α: forward prediction error
    - r_c: reflection coefficients
- Function CD1: is used for calculations of cepstral distance between two spectra using cepstral coefficients.
  - $D = cd_1(c_1, c_2, p)$

        - c_1:vector of cepstral coefficients of the first spectrum (without *c[0]*)
        - c_2: vector of cepstral coefficients of the second spectrum (without *c[0]*)
        - p: number of cepstral coefficients.
- Function LOADBIN: is responsible for loading binary integer data files to vector. Data are normalized in <-*1,1*> range.
    - *y=loadbin(filename[,channels,len])*
        - *filename*: name of binary data file
    - Optional parameters:
        - *channels*: number of channels of signal
        - *len*: desired length of ALL data
- Function MEL: conversion from linear frequency into mel-frequency scale.
    - *melf=mel(f)*
    - Input parameter:
        - *f*: frequency in Hz (vector or scalar)
    - Output parameter:
        - *melf*: frequency in mel (vector or scalar)
- Function SAVEBIN: Saving vector as INTEGER binary data file.
    - *savebin(filename,x)*
    - *filename*: desired name of binary data file
    - *x*: vector with data to saving

3.2. Specific Issues for Continuous Processing

Initially, the project aimed to achieve an algorithm that could apply continuous VAD. This means that an algorithm must be continuously recording an audio signal. This signal would be the entrance to the Voice Activity Detection algorithm.

By having a continuous signal of non-specific length, it should be a continuous signal processing. After obtaining the signal to be processed, we apply the algorithm to detect the areas that are activity voice detection and which are not, based on the algorithm described in the previous section.

When the VAD result is ready, the next step is to apply the start-and end-point detection. We take into account that parameters have been set for this part of the algorithm and make the necessary decisions to detect the point of beginning and end of each voice part of the signal.

Finally, the last requirement for the detection of activity is to save the parts of the signal that are voice activity detection in a buffer and discard the other.

As already stated, this project was designed for online data processing, so the entire process should be done online.

To fulfill the requirements of the project, we had to test several options that are described below.

- *Waverecord*: records the sounds using the Windows audio input device. The user can specify the number of samples (N), sample rate (fs), and the number of channels (CH). When we tested this function, we found that it is mandatory to specify the duration (in terms of the number of samples) of the signal to be recorded. A negative point was that every time one uses waverecord, this information is removed, so it does not allow continuous recording.
- *Audiorecorder:* works similarly to *Waverecord*, but it allows the user to save information in an object that is being recorded, thus allowing access to the data at any time. The user can also change the properties to suit the work. We provided access to information through the object, but the program would not let us process the data until it had finished recording. Thus, continuous processing of the data was not possible.
- *Data Acquisition Toolbox:* allows the user to select the audio input device used for data acquisition. With it, the user can get the audio signals during the specified time and then can process them. But as in the previous cases, it is necessary to obtain the data and then process it, so the toolbox does not provide the initial solution to the problem.

Because of these limitations, we looked for an alternative to meet the initial requirements of the project.

The solution was proposed to record an fixed-time audio signal and save it to a file.

- Once this action is completed, the two processes can be started simultaneously. On the one hand, the user can record a new signal file, which would resolve the loss of information from the input audio signal because the computation time is very low (0.011 seconds).
- On the other hand, the program could start processing the information from previously saved file at the same time.

We determined to set data process in frames of 256 points, and to read 128 new samples each time. First, we had to open the required file and then start reading until we reached the end of the cycle.

Latency is the time necessary to record the first piece of signal (for the demo, application is 8 ms signal) and save the file; computation time for the first piece of signal should be taken into account. Thus, latency of the process is:

$$\text{latency} = \text{t}_\text{s} + \text{t}_\text{r} + \text{t}_\text{c}$$

where:

- t_s: signal established time (8 seconds for demo application);
- t_r: recording and saving time: 8.3 seconds (8 seconds relatives to t_s);
- t_c: 11 ms.

This makes it possible to make a continuous recorded audio signal and at the same time process the signal, which would result in the application of VAD algorithm into the continuous signal required for the project.

3.3. Analysis of Speech Processing

As noted, the first step is to record the audio signal to save it as a file. To make this recording, we used the waverecord function because it involves little computation time and therefore can record continuously.

Once the file was ready, we read it in the manner described above to carry out the speech processing. Each time there is an interaction in the loop to read the full signal, it will obtain a vector with full signal and the other with the output of the VAD (outsig). The latter signal consists of frames of 32 ms. Once the loop has been performed completely, it will construct a new vector outsig with frames of 0.25 seconds, so outsig 15 frames constitute a new frame of outsig2. This new signal is used to process the start-and-end point detection. When this process is finished, this new signal will be saved in the buffer, and the process begins again.

3.4. Target Application Implementation

We have developed an application for showing the performance of the project graphically. This application will be based on applying an audio signal to the Voice Activity Detection algorithm. The results will show the start-and-end point detection to cut the signal. As a final step, an integrated digit recognizer has been added.

All results are displayed separately to better illustrate how the whole project has been developed.

3.4.1. Voice Activity Detection Algorithm Implementation

As a major part of the application, the VAD algorithm has its own implementation. Its operation has already been described. Its fundamental principles are based on the cepstral distance and adaptive threshold, as shown in Figure 13.

The starting point of this algorithm is analysis of the two first frames, Then, we discuss the entire cycle in terms of these frames. This computes the ambient noise, and it can detect which parts of the audio signal voice activity exists and which does not. The next step is to analyze a cycle in which the cepstral distance and adaptive threshold for the entire signal are calculated and then we determine, based on a pre-established criterion, which frames are active and inactive.

Figure shows the input audio signal and the cepstral distance threshold, and finally the voice detection expressed in frames within 0 and 1 sequence.

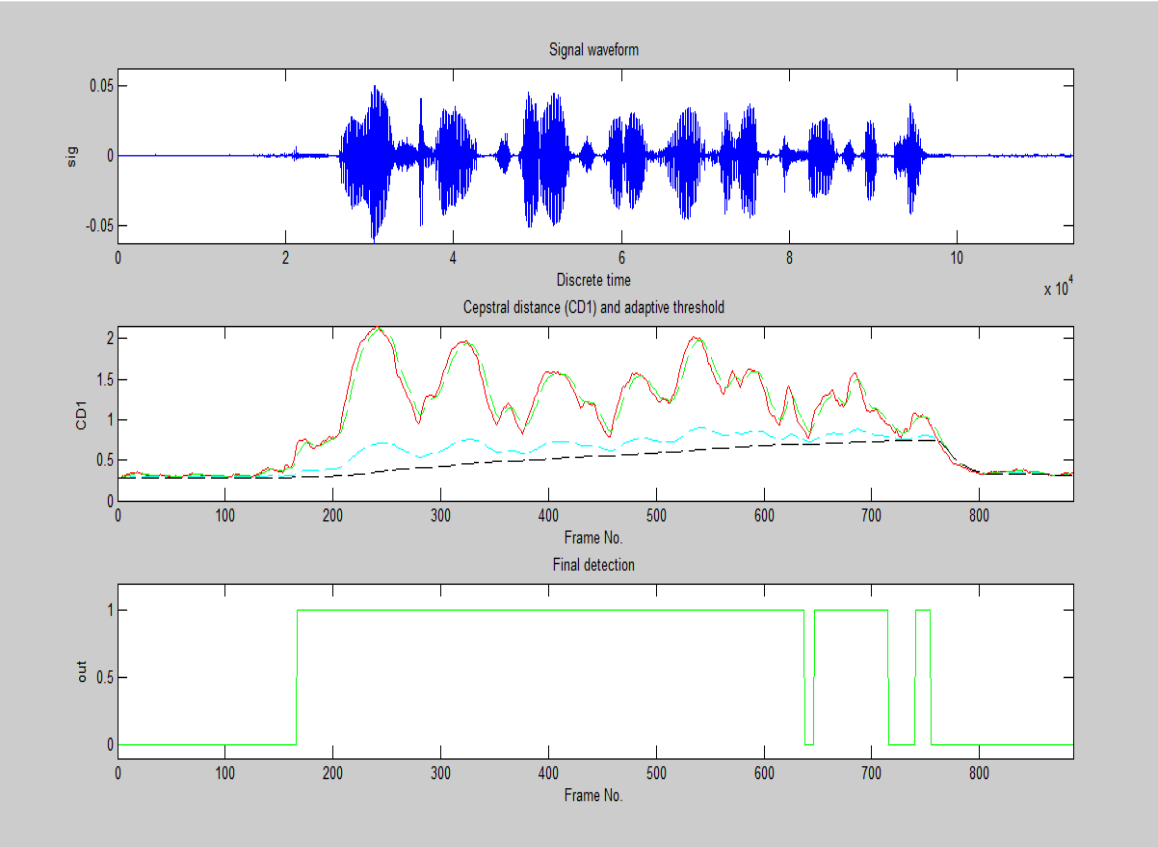


Figure 13. VAD implementation.

3.4.2. Start-and-End Point Detection Implementation

The purpose of start-and-end point detection is to detect when a signal starts and to identify just the active part of an audio signal. To make these decisions, we need a previously set criteria with which to optimize the results. Many parameters can be varied to get the best results.

The first problem posed by this part of the project is that the VAD output signal consists of frames of 32 ms, which is considered too short a time to implement the start-and end-point detection. We decided to build new frames of 15 VAD output signal frames each.

The next step is to decide how to determine if a frame is the beginning of the signal. We considered several options (described in the next chapter), but finally decided that the start sequence is marked by the end of 0111 and 0011.

The final step is to determine the range of samples to be taken to ensure a margin of error. We decided to take 1000 samples for the beginning and 2500 for the end. The reasons for these decisions will be explained in the next chapter. Figure 14 shows the final results.

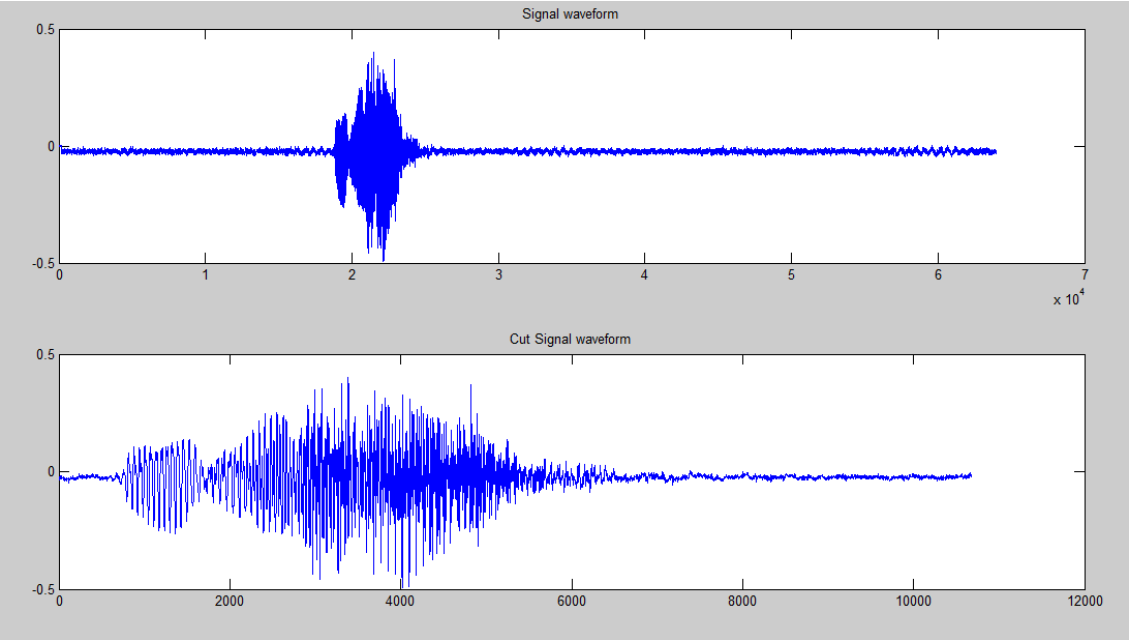


Figure 14. Start-and-End point detection implementation.

Figure 14 shows the output of the demo application. The first signal is the original 8 second-audio signal recorded to process with our final algorithm. The second signal is cut signal waveform after applying the start-and-end algorithm, resulting in a 1.35 seconds audio signal comprised of only the voiced part of the original signal.

3.4.3. Digit Recognizer

To give added value to the project, we decided to incorporate a simulation of speech recognition. This application is one of the most common in speech processing.

The application consists of a single digit recognizer between 0 and 9. It is needed to simulate the keypad of a mobile phone.

The main basis of this recognition is the previously recorded voice patterns for each digit. Using the DTW algorithm, we obtained a measure of similarity between the input audio signal and each of the patterns that had previously been recorded. The digit containing the audio signal is used to calculate the distances and the shortest distance.

Figure 15 shows how the digit recognizer is integrated into the final application.

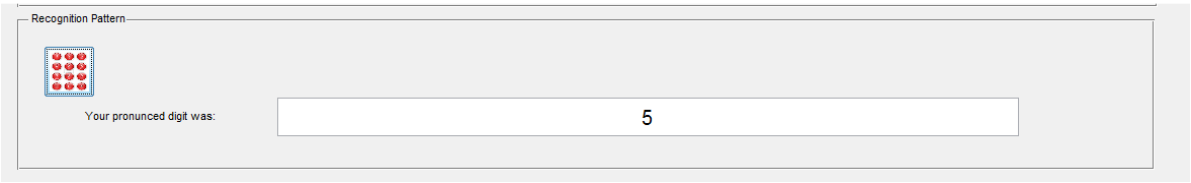


Figure 15. Digit Recognizer.

3.4.4. Final Application Implementation

New figure shows the final application:

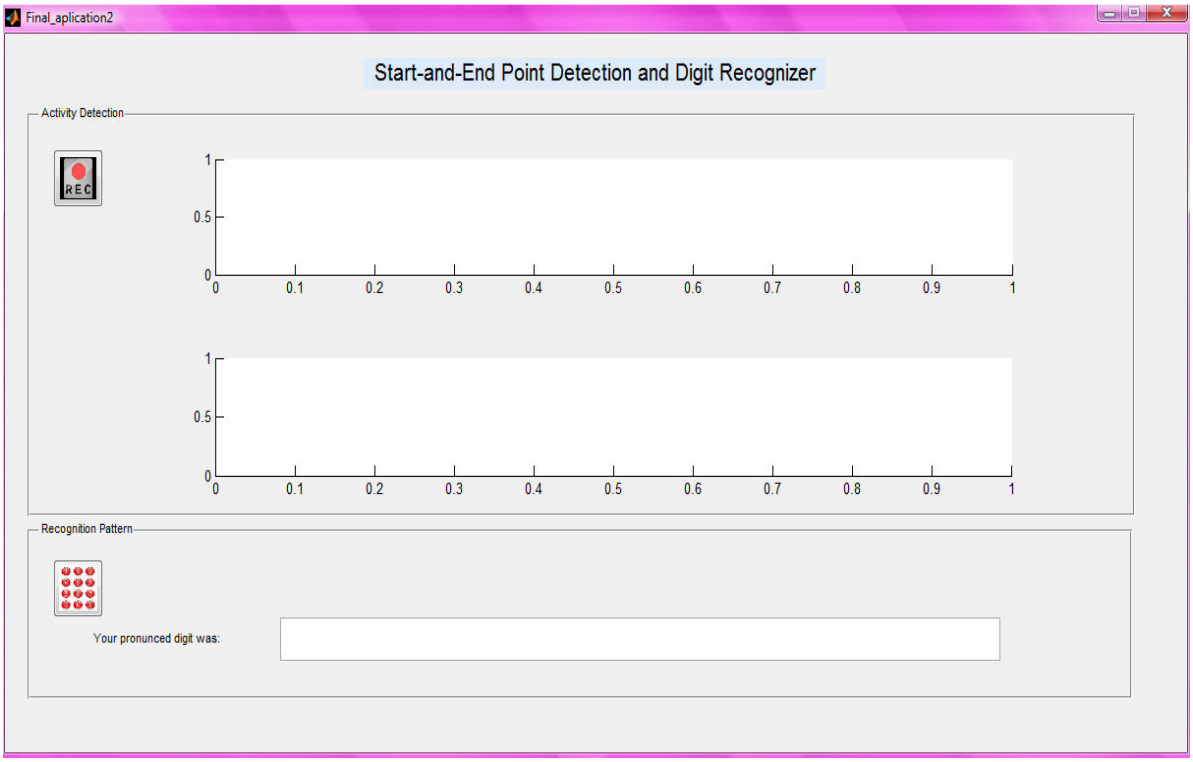


Figure 16. Final application.

The implementation of this application has been created entirely in MATLAB. It has been developed using a tool called GUIDE, an interface that simulates the development environment for deploying applications in MATLAB and Simulink.

It is a simple application that shows the results of the project graphically. It consists of two parts, the activity of the voice and speech recognition. It has two push buttons that are responsible for running both sides.

The voice activity detection portion has two graphics showing the original audio signal and cut signal after applying the VAD and start-and-end-point detection algorithm. In the speech recognition, a display shows the digit that has been spoken.

Figure shows 17 the final result after running the application.

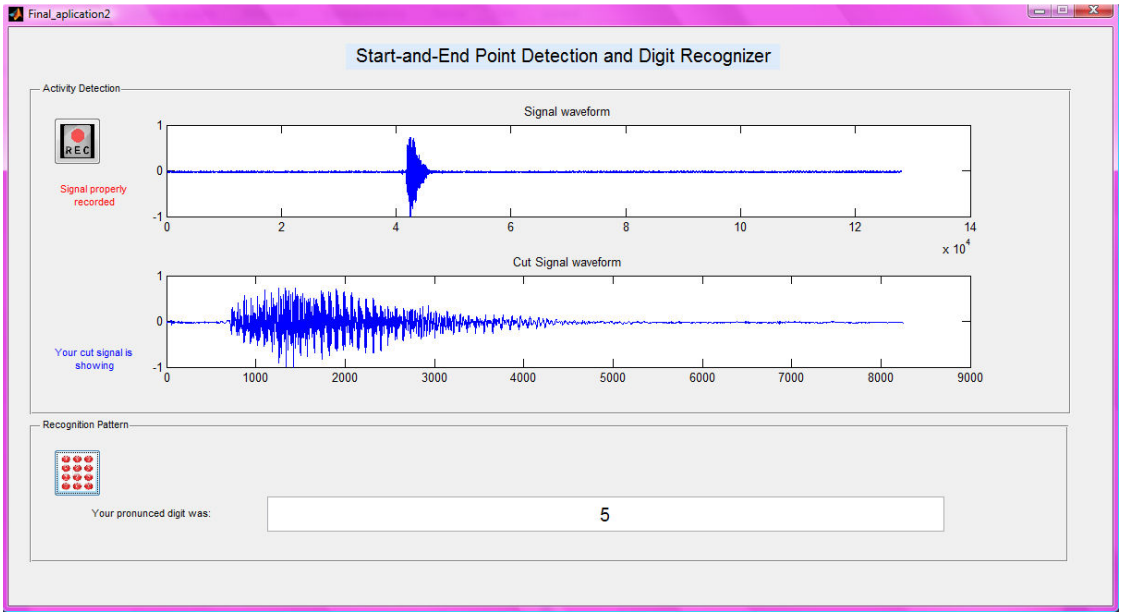


Figure 17. Performance application.

The application is operated via these steps. By activating the record button, the user should record a signal of 8 seconds containing at least one digit from 0 to 9. After confirming that the signal has been recorded correctly, a display appears that informs the user of that fact. Otherwise, the application will ask the user to repeat the recording.

The next step is to process the signal. This opens the file where the input signal has been recorded and it starts processing as has already been described. Once it identifies the start and end point signal, the result is displayed with the portion of the signal containing the digit.

If it is a signal that contains no voice activity, i.e., containing only noise. the application asks the user to repeat the process.

The other part of the application, the digit recognizer, is operated separately, as described above. If we recognize the digit that has been pronounced, we simply press the application button and it will show that digit.

The application has an approximate computation time for an 8-second audio signal:

- **activity voice detection:** 1.87 seconds;
- **digit recognizer:** 2.8 seconds.

The current design of the application can experience problems such as the detection of the voice at the very beginning of a file. This depends on the position of the active signal. This could be a problem if the active period takes place before the additional point that sets the default algorithm as the start

signal (1000 samples). This means that if the active signal begins in sample 100, the algorithm cannot detect the onset of this signal.

A possible improvement for such a situation would be to introduce an initial 8-second signal with no active part, containing only noise, when storing the continuous signal. The next signal would be the first to be processed with the start-and-end point detection algorithm and would be stored with the previous signal, and so on. Frames in which the beginning and end of the signal could not be detected would be stored intact. Therefore, frame n becomes frame n +1 of the newly stored signal. This new signal may be processed again with the algorithm to discard the non-active parts of the signal with a frame size multiplied by 2. This new process would solve the initial problem. Other possible situations could be resolved if the digit is between 2 files or the end of a file.

CHAPTER 4. TESTS AND RESULTS

This chapter will describe all the tests made to verify the correct operation of the application. We tested the algorithms and the application performance. The chapter will also include statistics of the results to draw the final conclusions of the project.

4.1. Tests

Tests in this chapter are purely illustrative. Many figures are shown to illustrate the variation in results and to indicate why we chose specific values for each parameter. The final choice of values was much more complex than will be illustrated in this report.

4.1.1. VAD Performance Tests

- The first parameter in the algorithm with large variations in results is q, the forgetting factor for background cepstrum forgetting.

  This factor influences mainly the detection of the next frame when the frame that is currently being assessed does not show voice activity. The greatest variations in results can be seen in the output of the start-and-end point detection, because if the sample is not properly adjusted, we cannot successfully complete the end point and there will be too many samples at the end of the clipped signal. The chosen value for the final implementation is 0.999.

  In terms of formulas and their dependence on the final algorithm, the choice of output signal affects the cycle of each frame.

```
%Cycle
[newdata, count]= fread(F, wlen/2, 'int16');
frame=[olddata;newdata];
  ai=burg(frame,cp);
  ci=a2c(ai(2:cp+1),cp,cp);

  D=pp*D+(1-pp)*sqrt(sum((c0-ci).^2));
  Dv=D;

  if ( D > Dmax )
    Dmax = qmax1*Dmax + (1-qmax1)*D ;
  else
    Dmax = qmax2*Dmax + (1-qmax2)*D ;
  end

  if ( D < Dmin )
    Dmin = qmin1*Dmin + (1-qmin1)*D ;
  else
    Dmin = qmin2*Dmin + (1-qmin2)*D ;
  end

  Dyn=Dmax-Dmin ;
  Dp=Dmin+perc/100*(Dmax-Dmin);
  Dpv=Dp;
  Dmaxv=Dmax;
  Dminv=Dmin;
```

```
if ( D>Dp & Dyn > 0.20),
  out=1;
  okdata=[okdata;newdata];
  outsig=[outsig;out];
else
  out=0;
  outsig=[outsig;out];
  c0=q*c0+(1-q)*ci;
end;
 signal=[signal;newdata];
 olddata=newdata;
 end;
 %End cycle
```

- The following parameter affects whether a frame is active or inactive. The result is reflected in the blue line in the following figures. If the respective red line is greater than the blue one, the frame is marked as active.

  ```
  Dp=Dmin+perc/100*(Dmax-Dmin);
  ```

  Figures 18 value show how this affects the final result set of VAD.

  - Perc=20:

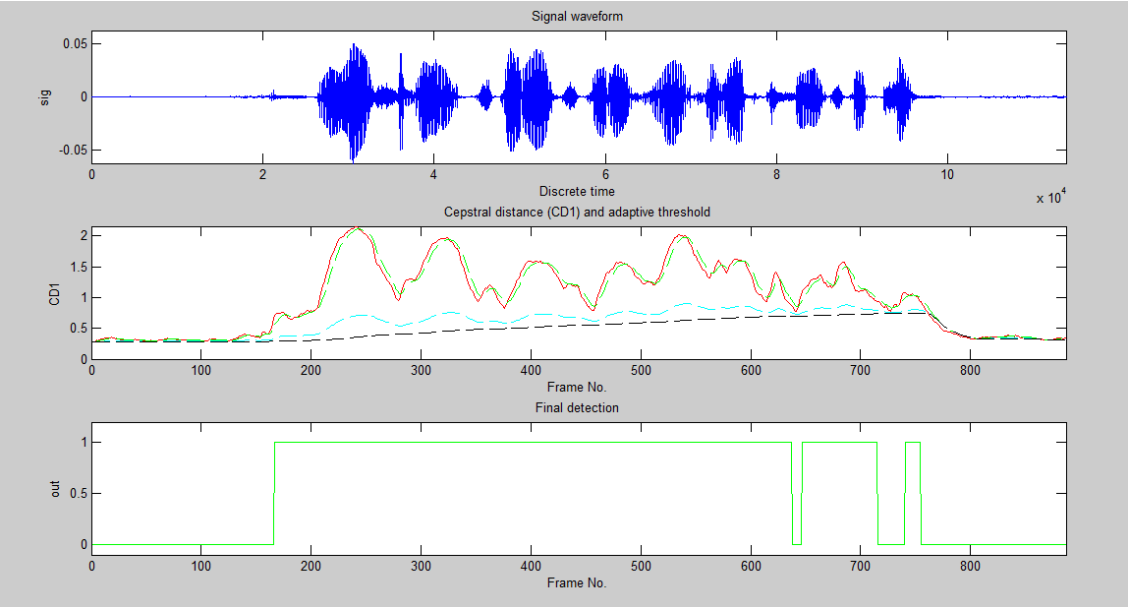


Figure 18. Percentage=20.

- Perc=50:

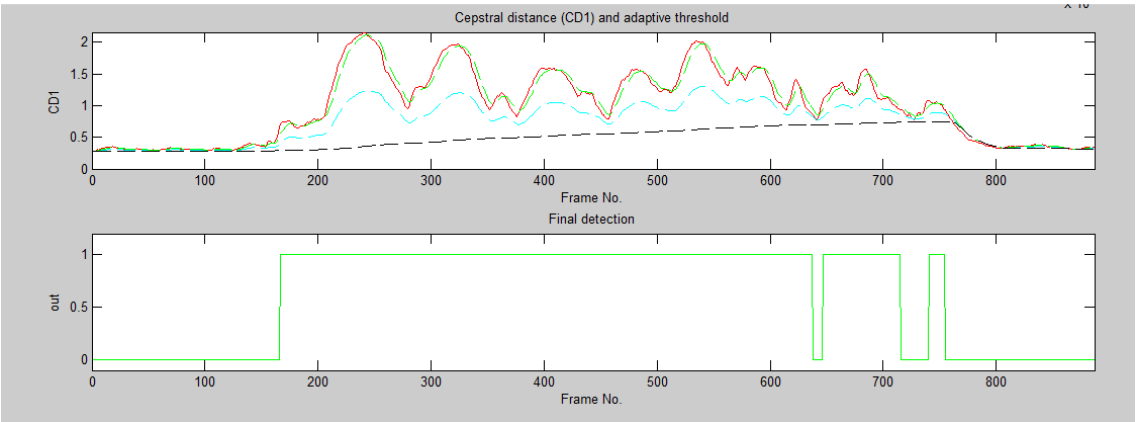


Figure 19. Percentage =50.

- Perc:80

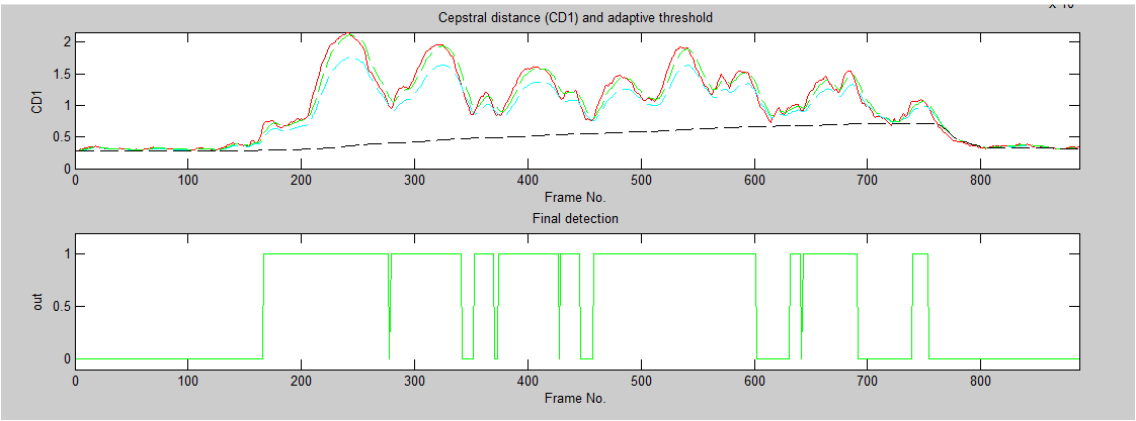


Figure 20. Percentage = 80.

The final value chosen is 20, since it achieved the best results.

- The next parameter is compared with the previous one. This is indicated by the red line, which shows results obtained for different values. Its optimal value is 0.86.

```
D=pp*D+(1-pp)*sqrt(sum((c0-ci).^2));
```

- pp=0.6

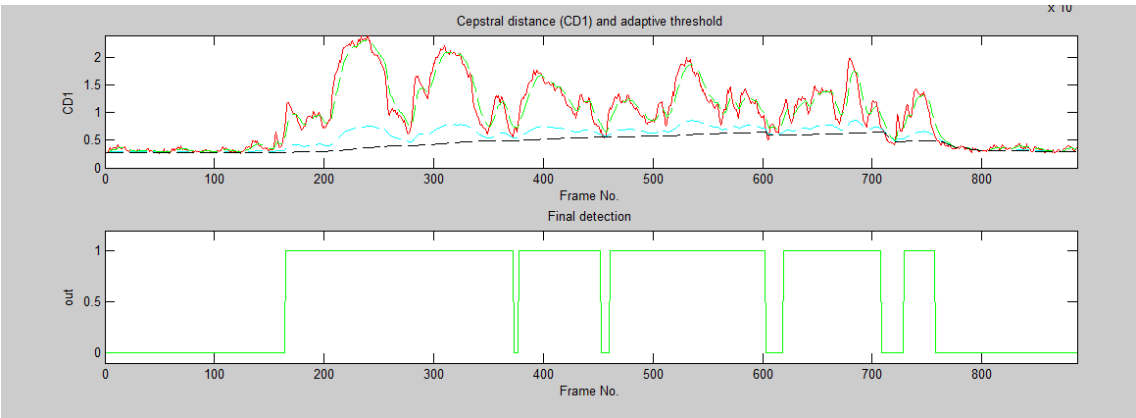


Figure 21. pp= 0.6.

- pp=0.99

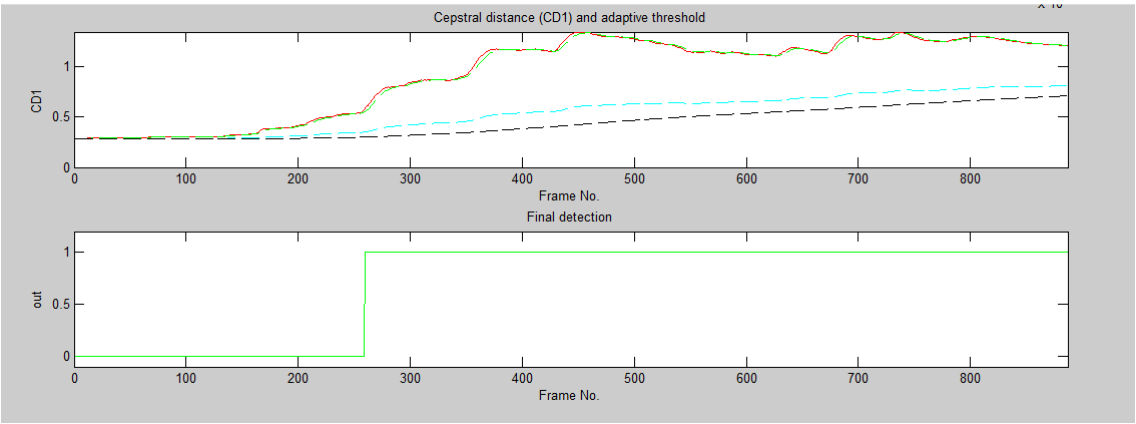


Figure 22. pp=0.99.

- The following parameters are the forgetting parameters of maximum and minimum. They are used to calculate the minimum and maximum range of each frame (black and green lines respectively) and to calculate the difference between them that if greater than 0.25 implies that the frame is active.

```
if ( D > Dmax )
    Dmax = qmax1*Dmax + (1-qmax1)*D ;
  else
    Dmax = qmax2*Dmax + (1-qmax2)*D ;
  end

  if ( D < Dmin )
    Dmin = qmin1*Dmin + (1-qmin1)*D ;
  else
    Dmin = qmin2*Dmin + (1-qmin2)*D ;
  end
```

  The chosen values are:

  - qmax1=0.7: forgetting parameter of maximum distance (in case of increase)
  - qmax2=0.95: forgetting parameter of maximum distance (in case of decrease)
  - qmin1=0.995: forgetting parameter of minimum distance (in case of increase)
  - qmin2=0.9995: forgetting parameter of minimum distance (in case of decrease)

  Figure 23 shows the result for the chosen values:

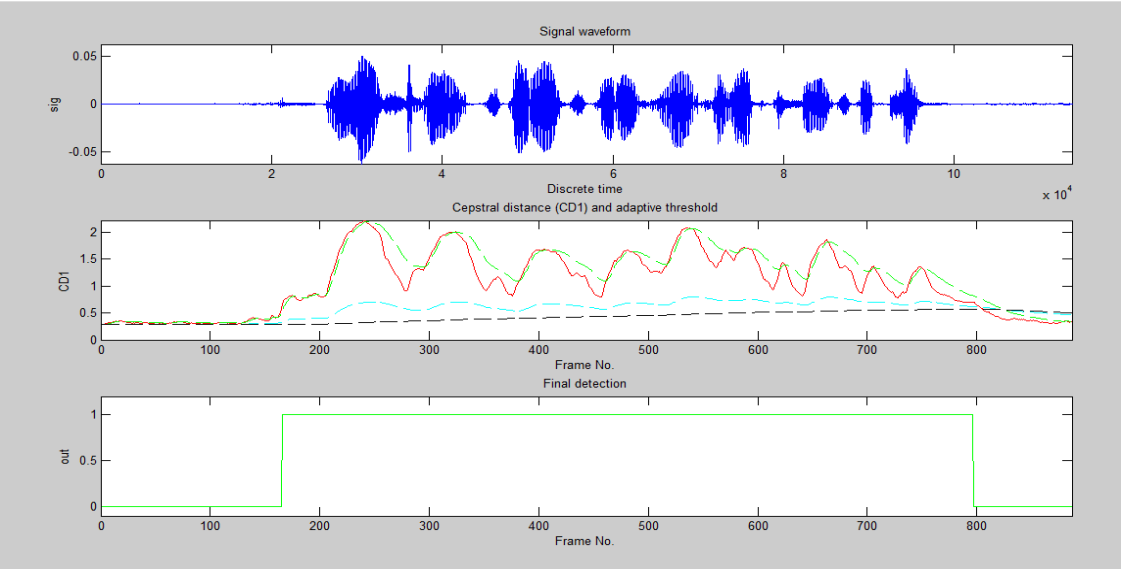


Figure 23. Forgetting parameters for optimal results.

These values should be between 0-1, so it was interesting to see the extreme situations when we were checking the influence of these parameters when these values are almost 1 and how that influences into the final results of VAD:

- qmax1=0.999

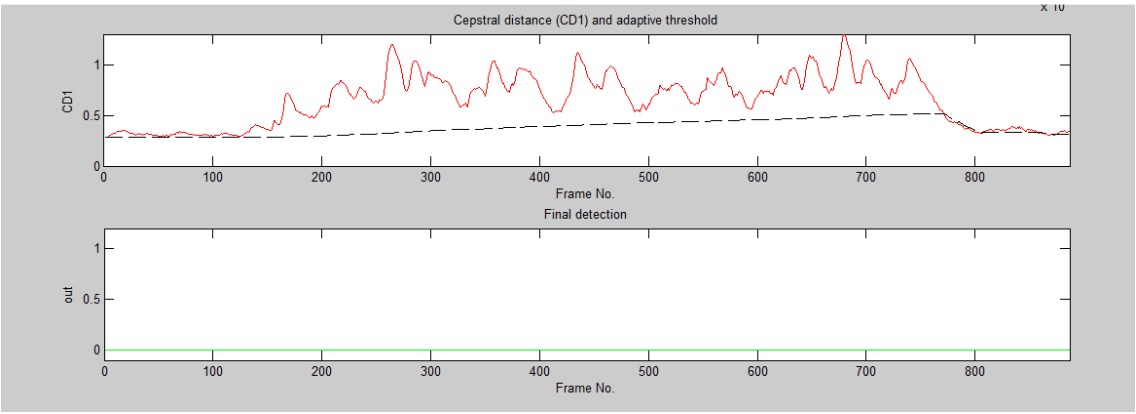


Figure 24. qmax1=0.999.

- qmax2=0.999

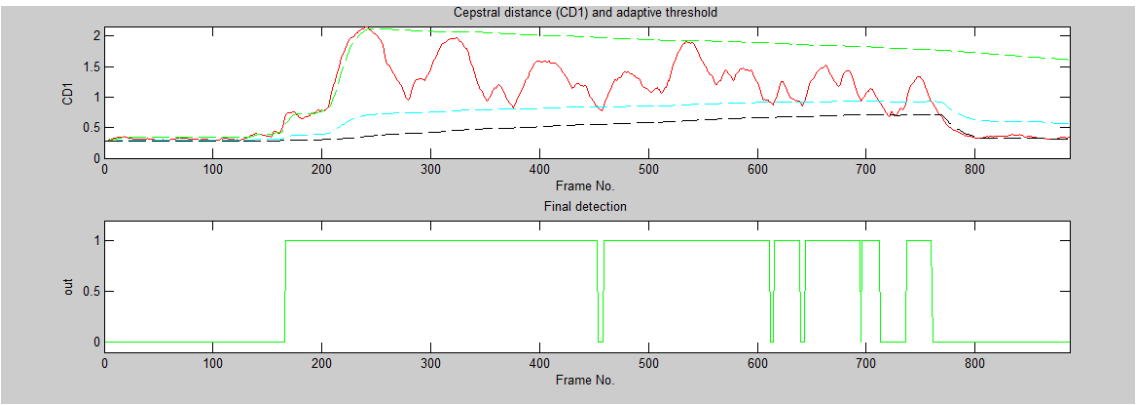


Figure 25. qmax2=0.999.

- qmin1= 0.999

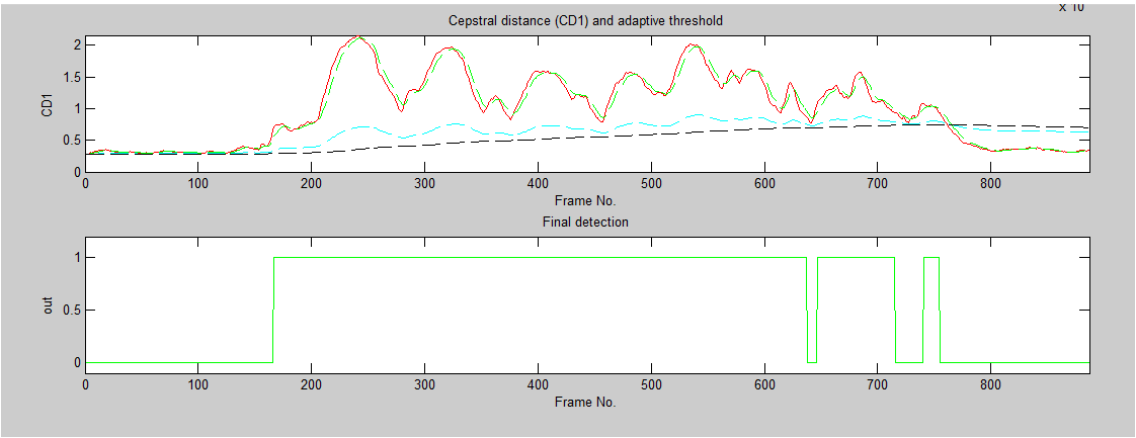


Figure 26. qmin1=0.999.

- qmin2= 0.95 (test from 0.795)

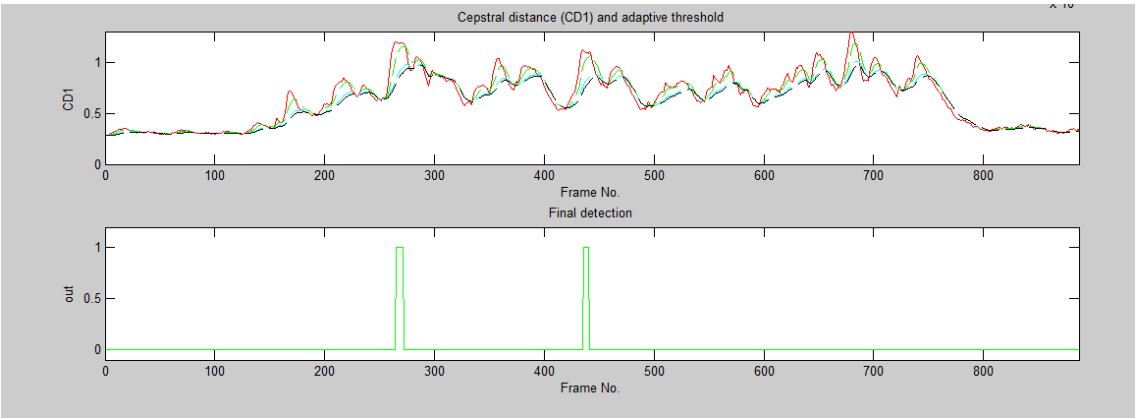


Figure 27. qmin2=0.95.

4.1.2. Start-and-End Point Detection Tests

The tests in this section are primarily based on three main parameters.

- The parameter is responsible for whether one frame formed by other frames with different values of 0 and 1 should be finally identified as active or inactive.

```
i=1;
outsig2=[];
while i<= wnumf,
    if mean(outsig(i:i+14))<0.3,
        newframe=0;
        outsig2=[outsig2;newframe];

    elseif mean(outsig(i:i+14))>0.3,
        newframe=1;
        outsig2=[outsig2;newframe];
       end;
        i=i+15;
end;
```

Final choice is: 0.3.

- If mean>0.3:

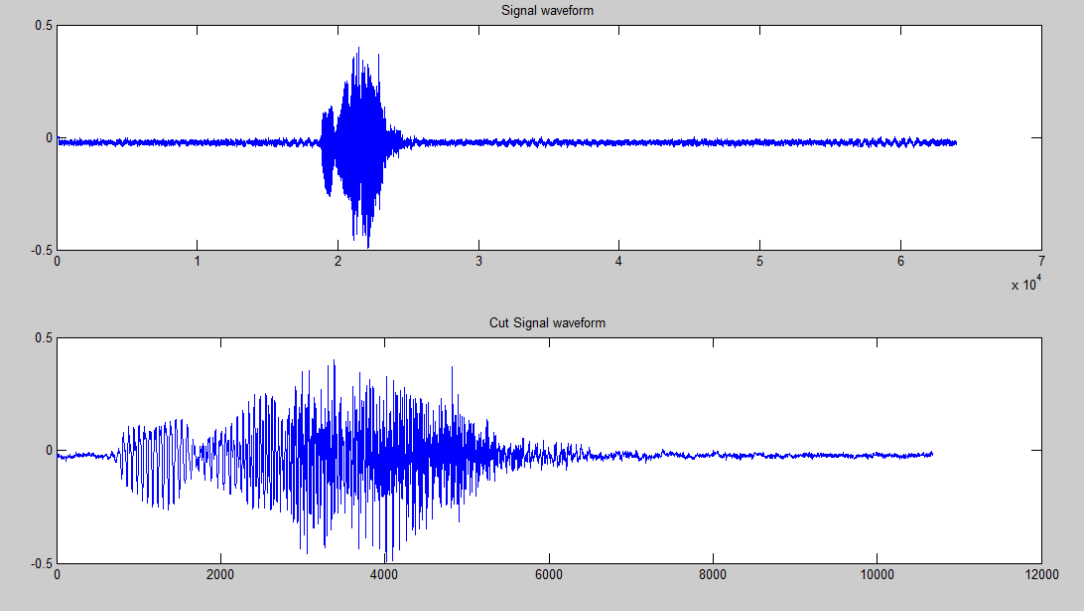


Figure 28. mean>0.3.

- Mean>0.7

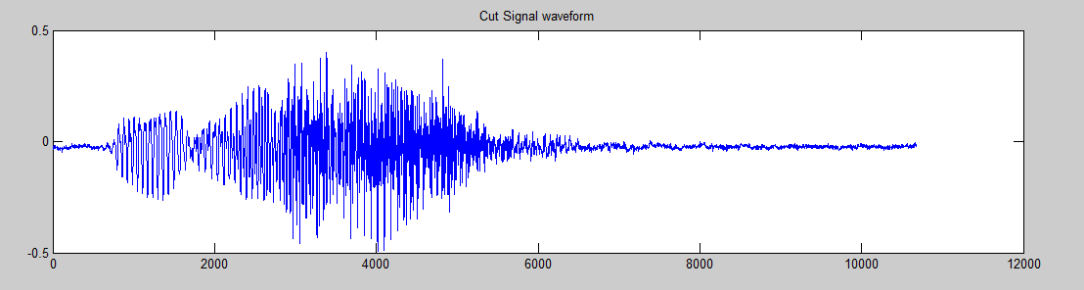


Figure 29. mean>0.7.

- Mean>0.9

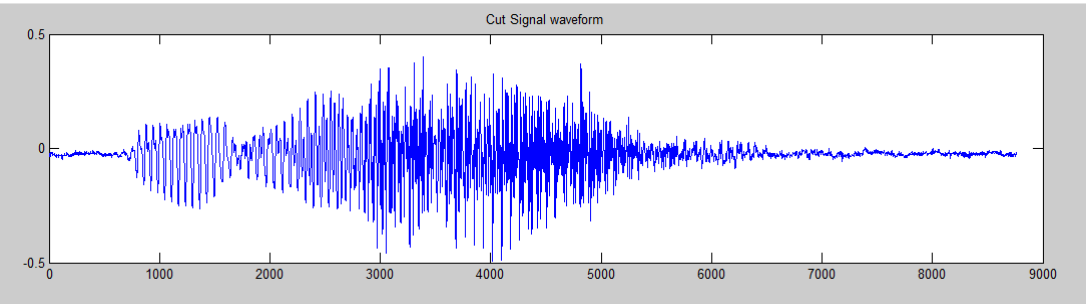


Figure 30. mean>0.9.

- Another important decision is related to 0 and 1 sequence:
  - 000111: there were problems with the signal that started close to 0;
  - 111000: there were problems determining which signal finished close to the end of the recorded signal.

```
%Definition beginning and end signal
for e=2:(l_o2-2),
    if outsig2(e)==1 && outsig2(e+1)==1 && outsig2(e+2)==1 &&
outsig2(e-1)==0 ,
        begin=(e-1)*128*15;
        beginmat=[beginmat;begin];

    end;
    if outsig2(e)==0 && outsig2(e+1)==0 && outsig2(e-1)==1 &&
outsig2(e-2)==1 ,
        ends=(e-1)*128*15;
        endsmat=[endsmat;ends];

    end;
end;
```

- The last parameter that influences the final results is the number of margin samples requires to assure that the clipped signal is completed:

```
if length(beginmat)>0,
%l_begin=length(beginmat);
begin=beginmat(1)-1000;
else
    set(handles.mostrar2, 'String', 'Please, record a new
signal')
    return;
end;
%l_ens=length(endsmat);
if length (endsmat)>0,
ends=endsmat(1)+1500;
else
    set(handles.mostrar2, 'String', 'Please, record a new
signal')
    return;
end;
```

  - Beginning:
    - 8000

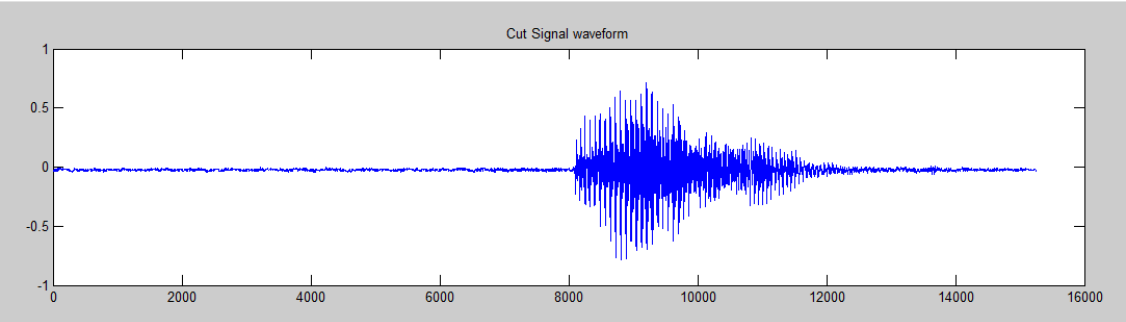


Figure 31. Beginning+8000 samples.

- 4000

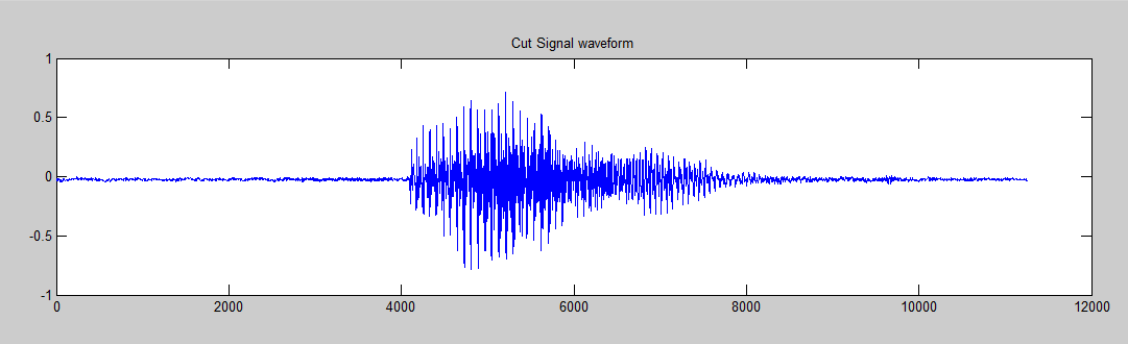


Figure 32. Beginning +4000 samples.

- 1000

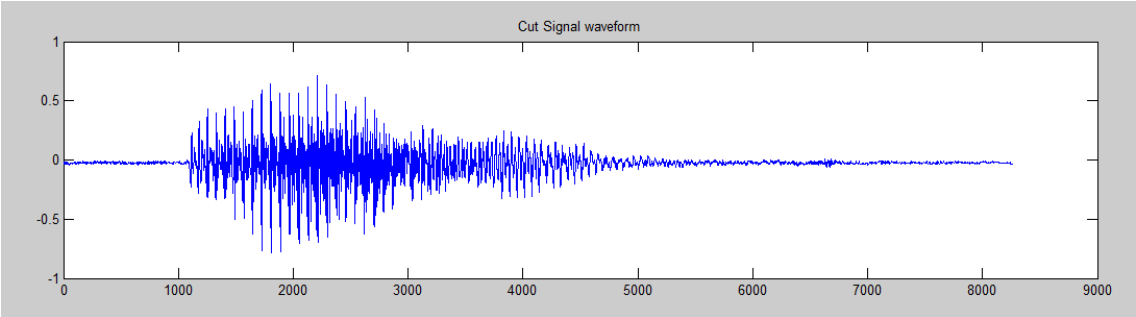


Figure 33. Beginning +1000 samples.

- End:
  - 16000

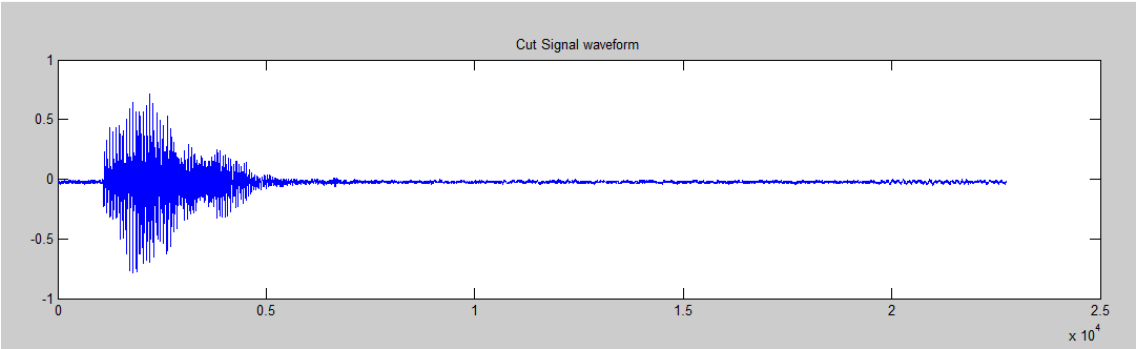


Figure 34. End + 16000 samples.

- 8000

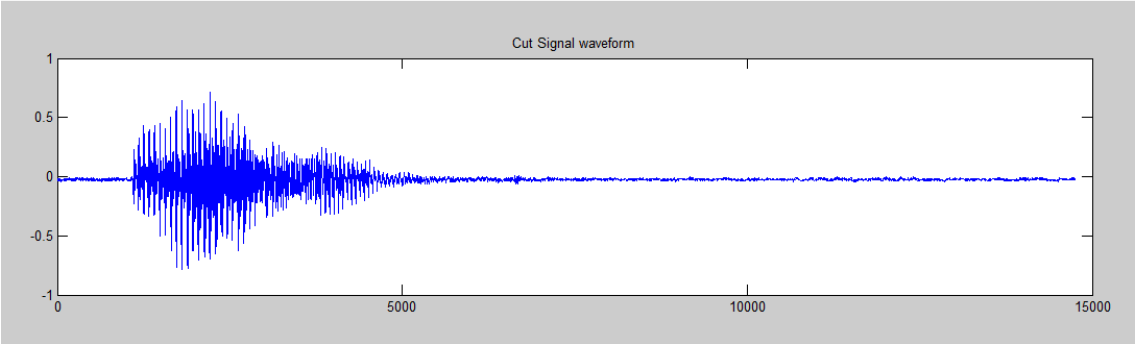


Figure 35. End + 8000 samples.

- 1500

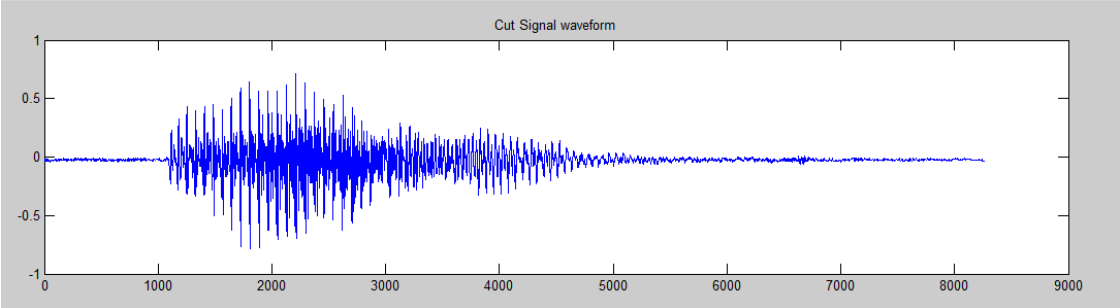


Figure 36. End+ 1500 samples.

The values chosen are 1000 samples for the beginning and 1500 for the end.

4.1.3. Digit Recognizer Tests

The evidence in this phase focused on two main elements.

- The first part of the tests was to identify the optimal microphone for proper functioning of the application. The microphone had to be able to record patterns so they were recognizable later. We tested the microphone that incorporates the PC and a Logitech headset. After several tests, we determined that the results were better with the microphone that incorporates the computer because it does not interfere as much with effects like breathing or pitch of the signal.
- Second, we identified the signal margin. The results are optimal for cases in which an input signal obtained was from the amplitude margins of 0.5 and -0.8.

4.2. Statistical Results

4.2.1. Activity Detection Performance Results

The option implemented return better results.

These tests have been done on the final algorithm and were carried out both manually and automatically. The idea is to manually check where the signal begins and ends and then check the same occurrences through the application.

The tests were performed in two environments, a loud noisy one and one that was quiet, to determine if differences were found in the results. The tests were repeated 40 times for both cases to obtain reliable results.

To show the results statistically, several tests have been made.

- These first results are in a quiet environment:

Table 1 shows some values for the beginning and end for each signal.

	Manual		Automatic	
#	Begin	End	Begin	End
1	7500	14700	2840	21200
2	9000	14000	9600	17360
39	15800	24200	14360	28880
40	24000	32000	23960	34640

Table 1. Example of start and end point of the tests.

In this series (see the appendix for the complex tables), the difference has been calculated manually and automatically to determine the beginning and end of each signal. Figures 37 and 38 illustrate the data.

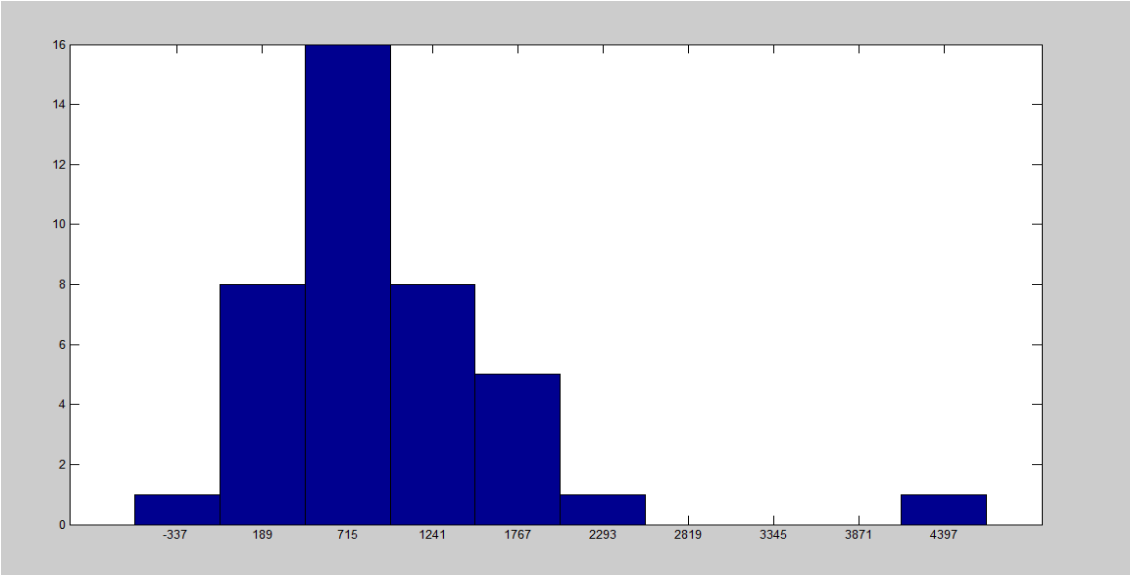


Figure 37. Histogram of beginning difference.

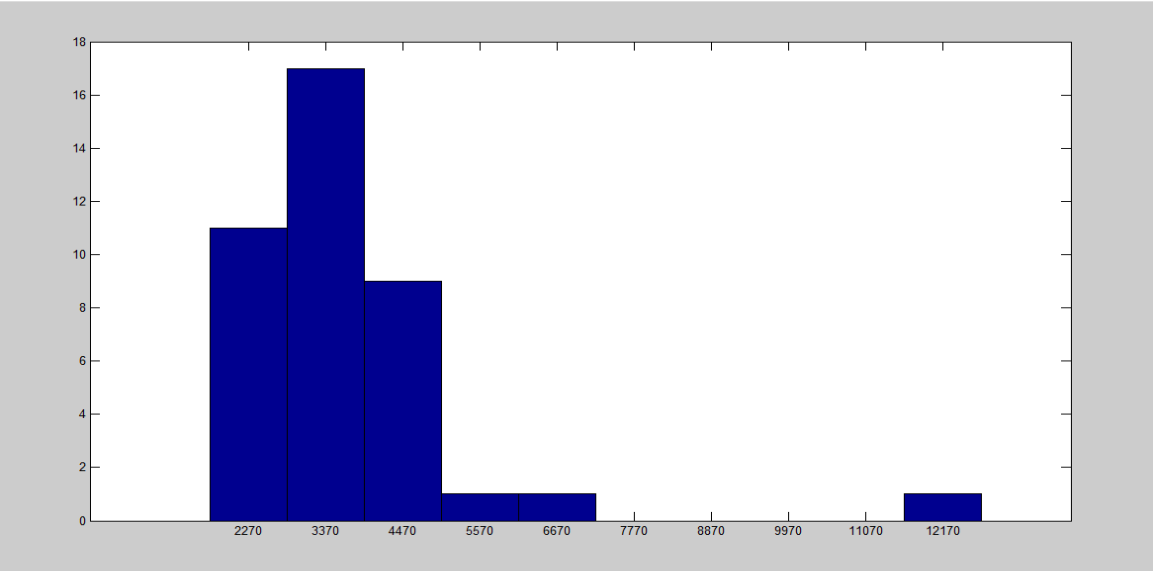


Figure 38. Histogram of ending difference.

Table 2 shows the relationship between the start and end point of the signal, as well as the differences, mean, and standard deviation for all samples.

	Manual		Automatic							
#	Begin	End	Begin	End	Δ Begin	Δ End	σ begin	Mean Begin	σ end	Mean End
1	7500	14700	2840	21200	4660	6500	820,448048	1025	1743,23	3731,5
2	9000	14000	9600	17360	-600	3360				
39	15800	24200	14360	28880	1440	4680				
40	24000	32000	23960	34640	40	2640				

Table 2. Non noise Start-and-End point detection tests.

- In a noisy environment:

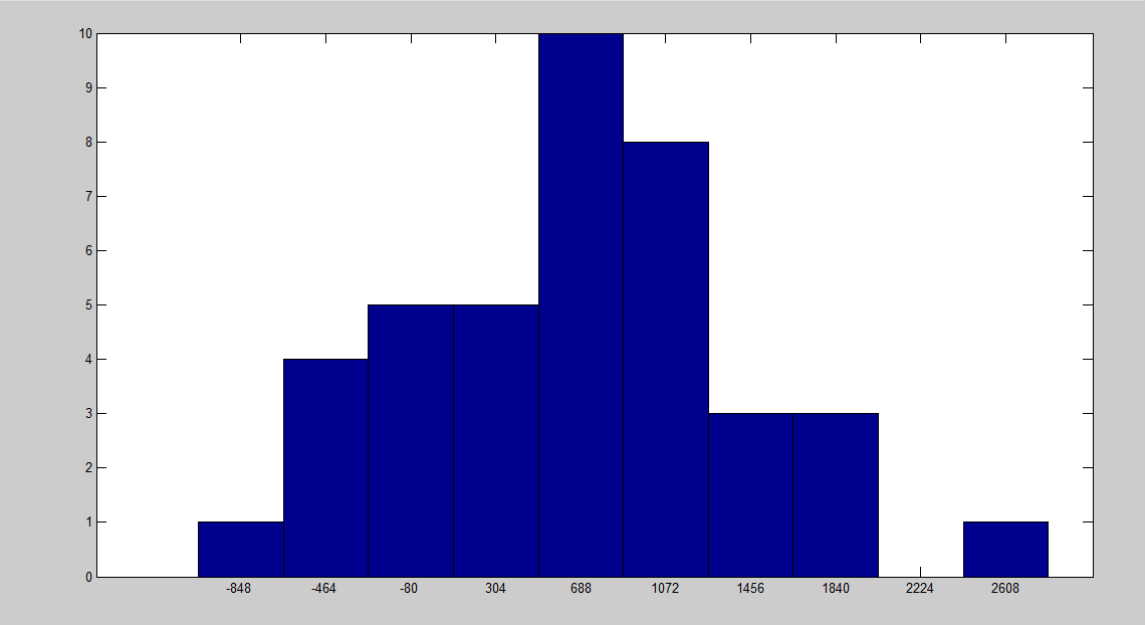


Figure 39. Beginning difference in noise environment.

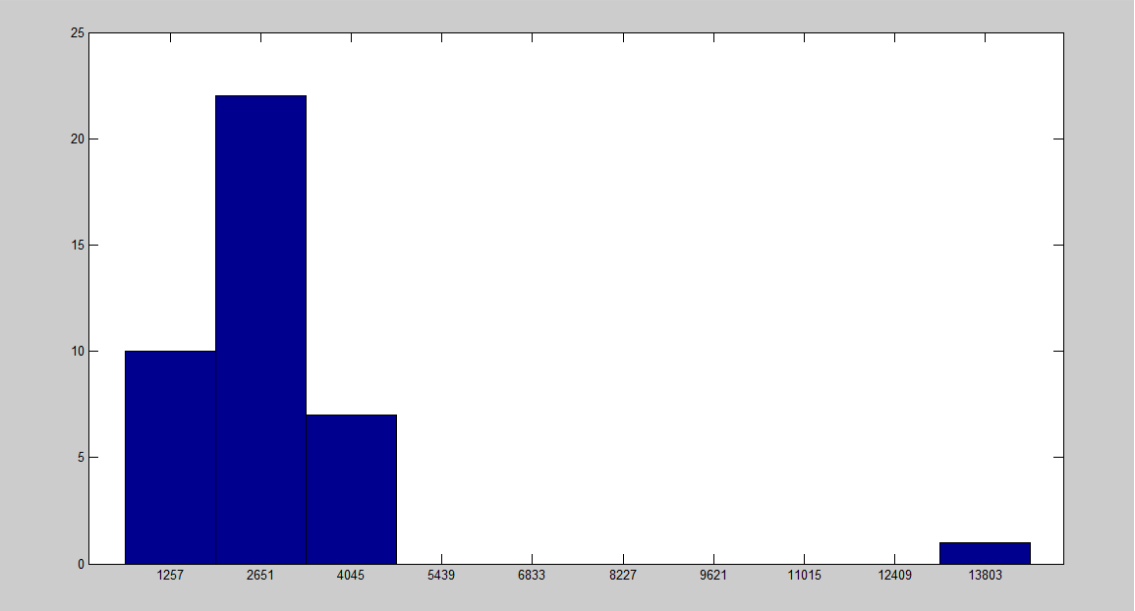


Figure 40. End difference in noise environment.

	Manual		Automatic							
#	Begin	End	Begin	End	Δ Begin	Δ End	σ begin	Mean Begin	σ end	Mean End
1	10900	19000	10520	23120	380	4120	752,80	678	2088,7	2838,5
2	10800	16700	10520	19280	280	2580				
39	25000	31000	23960	34140	1040	3140				
40	16000	25500	14360	28380	1640	2880				

Table 3. Noisy environment Start-and-End point detection tests.

The general conclusions are that to detect the end point, more samples have to be used for the beginning point. This is because the start-and-end point detection algorithm is designed with a greater insertion of samples at the end of the detected signal to provide a greater margin and avoid leaving the signal incomplete.

On the other hand, we achieved very similar results in both media, so it can be said that the algorithm is independent of the conditions in which it is used.

4.2.2. Digit Recognizer Results

Another way to check the algorithm and the expected results is to use the complete application, which recognizes the digit containing the signal. Although this is not the main purpose of the project, it is part of the final application. Correct recognition also confirms that the algorithm is functioning properly.

We tested digit recognizer by testing each of the numbers to be recognized 10 times and then examining the results. Tests have been performed for 3 different speakers, and each has made a series of 10 samples for each digit. Table 4 shows the average results.

Number	Success	Failure	Percentage
0	10	0	100
1	8	2	80
2	8	2	80
3	7	3	70
4	10	0	100
5	8	2	80
6	10	0	100
7	9	1	90
8	10	0	100
9	10	0	100

Table 4. Digit recognizer statistics.

CHAPTER 5. CONCLUSIONS

As has been stated, the motive for this project was the implementation of an algorithm that can detect the beginning and end of voice activity in a signal.

The beginning of the project was based on a Voice Activity Detection algorithm. The main idea was to detect voice activity continuously. A signal is recorded continuously with the aim of analysing smaller sequences to determine whether there is voice activity. If activity is detected, this part of the signal is saved in a buffer. Otherwise, it is discarded.

An application was developed to show the work of the project graphically. It is based on the algorithm for the clipped signal after performing speech recognition using a simple algorithm that allows recognition of the pronounced number.

The major contributions of my work can be summarized as:

- studying and understanding the operation of the Voice Activity Detection algorithm. Once it was understood, the next step was to adapt the algorithm and particularize it to obtain desired results for my project;
- once the results of the VAD algorithm are ready, I designed a start-and-end point detection algorithm that can detect the beginning and end of the signal activity. An algorithm has been designed to optimize the results and allow us to obtain from the output a signal in which only the remaining part of the signal showed detected activity;
- to show the results of the project, I implemented an application entirely in MATLAB. In this application, I integrated the VAD and the start-and-end point detection algorithm, and as a complement, I created a demonstrative simple digit recognizer based on a simple DTW algorithm;
- the last step of the project has been to test the performance of the algorithm. Tests have been conducted in different environments to ensure that it operates independently of conditions.

It is possible to say that work achieved the initial goal of the project, which was to develop an algorithm capable of returning a signal in which there is voice activity. It should be added that future work on this topic could focus on the possibility of using the Data Acquisition Toolbox for online listening and recording of data and further processing, without having to record small sequences of the voice as this project did.

Bibliography

[1] Laver J., Principles of phonetics, Oxford University Press., Oxford, UK (1994).

[2] E. G. Schukat-Talamazzini. Automatische Spracherkennung. Vieweg Verlag (1995).

[3] J. G. Proakis and D. G. Manolakis. Digital Signal Processing: Principles, Algorithms, and Applications. Upper Saddle River, NJ: Prentice-Hall (1996).

[4] L. Rabiner and B. H. Juang. Fundamentals of Speech Recognition. Englewood Cliffs, NJ: Prentice-Hall. (1993).

[5] J.R. Deller Jr., J.H.L. Hansen, J.G. Proakis, Discrete-Time Processing of Speech Signals, IEEE Press, New York. (2000).

[6] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, Discrete-Time Signal Processing, U. Saddle River, Ed. NJ:Prentice Hall. (1999).

[7] L.R. Rabiner and R.W. Schafer, Theory and Application of Digital Speech Processing, First Edition, Prentice Hall, New York. (2011).

[8] Doc. Ing. Petr Pollak. Speech Technology in Telecommunications slides. (2012).

[9] http://www.clear.rice.edu/

[10] M. Y. Appiah, M. Sasikath, R. Makrickaite, M. Gusaite, “Robust Voice Activity Detection and Noise Reduction Mechanism”, Institute of Electronics Systems, Aalborg University.

[11] Ramírez, J.; J. M. Górriz, J. C. Segura. "Voice Activity Detection. Fundamentals and Speech Recognition System Robustness". (2007).

[12] Matthaei, Young, Jones, Microwave Filters, Impedance-Matching Networks, and Coupling Structures McGraw-Hill 1964.

[13] M.J.L. DE HOON, T.H.J.J. VAN DER HAGEN, H. SCHOONEWELLE, AND H. VAN DAM, Why YULE-WALKER should not be used for autoregressive modelling

[14] Broersen, P.M.T. and Wensink H.E. (1993) IEEE Transactions on Signal Processing, 41, 194-204.Cybenko, G. (1980) Society for Industrial and applied Mathematics, Journal on Scientific and Statistical Computing.

[15] Priestley, M.B. (1994) Spectral Analysis and Time Series. Academic Press, London.

[16] Parzen, E. (1961) Technometrics.

[17] Taylor, Paul. 2009. Text-to-Speech Synthesis.

[18] Rabiner and Schafer, 2007,, Introduction to Speech Signal . The Netherlands.

[19] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=925340

[20] S. Young, The HTK Book: for HTK Version 2.1, Cambridge, England: Cambridge University Press, 1997.

[21] P. Pollak, P. Sovka and J. Uhlir, “Noise System for a Car”, proc. of the Third European Conference on Speech, Communication and Technology – EUROSPEECH’93, (Berlin, Germany), pp. 1073-1076,Sept. 1993.

Annexes

4.3. Tests

4.3.1. Algorithm tests

4.3.1.1. Non noisy environment

	Manual		Automatic			
#	Begin	End	Begin	End	Difference Begin	Difference End
1	7500	14700	2840	21200	4660	6500
2	9000	14000	9600	17360	-600	3360
3	17000	21000	16280	26960	720	5960
4	12500	16500	12440	21200	60	4700
5	24900	29400	23960	32720	940	3320
6	21900	28100	20120	30800	1780	2700
7	15800	23200	14360	26960	1440	3760
8	11000	17200	10520	19280	480	2080
9	17000	25100	16280	26960	720	1860
10	6100	13000	4760	15440	1340	2440
11	30000	40000	29720	42320	280	2320
12	13900	21200	12440	25040	1460	3840
13	15900	23000	14360	26960	1540	3960
14	23000	28500	22040	32720	960	4220
15	22900	31000	22040	34640	860	3640
16	29000	35000	27800	38480	1200	3480
17	33800	38100	33560	42320	240	4220
18	18900	28100	18200	32720	700	4620
19	10900	15100	10520	19280	380	4180
20	21100	30000	20120	32720	980	2720
21	23000	31000	22040	32720	960	1720
22	23000	32100	22040	34640	960	2540
23	27400	37000	25880	40400	1520	3400
24	27900	34800	25880	38480	2020	3680
25	35100	40000	33560	44240	1540	4240
26	24700	29100	23960	32720	740	3620
27	21000	28000	20120	30800	880	2800
28	19100	24100	18200	26960	900	2860
29	18800	27000	16280	30800	2520	3800
30	12800	26000	12440	28880	360	2880
31	17000	22000	16280	25040	720	3040
32	21000	30000	20120	34640	880	4640
33	21000	31100	20120	34640	880	3540
34	23000	31100	22040	34640	960	3540
35	12000	20000	10520	23120	1480	3120

36	20500	26100	20120	28880	380	2780
37	14800	20000	14360	32720	440	12720
38	25200	31500	23960	34640	1240	3140
39	15800	24200	14360	28880	1440	4680
40	24000	32000	23960	34640	40	2640

Table 5. Tests non noise environment.

4.3.1.2. Noisy environment

	Manual		Automatic			
#	Begin	End	Begin	End	Difference Begin	Difference End
1	10900	19000	10520	23120	380	4120
2	10800	16700	10520	19280	280	2580
3	11000	18000	10520	19280	480	1280
4	16800	24000	16280	26960	520	2960
5	18000	24100	16280	28380	1720	4280
6	15500	19100	14360	22620	1140	3520
7	25000	30100	23960	32220	1040	2120
8	22000	29200	22040	32220	-40	3020
9	18000	23500	18200	26460	-200	2960
10	10000	20000	8600	24540	1400	4540
11	23000	34000	22040	37980	960	3980
12	21000	28000	20120	30300	880	2300
13	12100	22000	12440	24540	-340	2540
14	16000	25500	16280	27260	-280	1760
15	21000	27000	20120	30300	880	3300
16	21000	27500	18200	42000	2800	14500
17	12000	18500	12440	20700	-440	2200
18	20000	30000	20120	32220	-120	2220
19	30000	37000	29720	37980	280	980
20	18000	28000	18200	30300	-200	2300
21	25000	35500	23960	36060	1040	560
22	27000	34000	25880	36060	1120	2060
23	22000	32000	22040	34140	-40	2140
24	19000	30000	18200	32220	800	2220
25	15000	22500	14360	24540	640	2040
26	21000	30000	20120	32220	880	2220
27	36000	43000	35480	45660	520	2660
28	23500	32500	22040	34140	1460	1640
29	24500	30500	23960	32220	540	1720
30	25500	37000	25880	37980	-380	980
31	16500	25500	16280	28380	220	2880
32	17500	25000	16280	26460	1220	1460
33	19000	28500	18200	30300	800	1800
34	18000	26500	16280	28380	1720	1880

35	21000	29500	22040	32220	-1040	2720
36	18000	24000	16280	28380	1720	4380
37	21000	27000	20120	30300	880	3300
38	29000	36500	27800	39900	1200	3400
39	25000	31000	23960	34140	1040	3140
40	16000	25500	14360	28380	1640	2880

Table 6. Tests noise environments.

4.3.2. Digit Recognizer tests

Digit	Result	OK
0	0	yes
0	0	yes
0	0	yes
0	0	yes
0	0	yes
0	0	yes
0	0	yes
0	0	yes
0	0	yes
0	0	yes
1	1	yes
1	1	yes
1	1	yes
1	1	yes
1	1	yes
1	9	no
1	1	yes
1	1	yes
1	1	yes
1	4	no
2	2	yes
2	2	yes
2	2	yes
2	0	no
2	2	yes
2	2	yes
2	4	no
2	2	yes
2	2	yes
2	2	yes
3	3	yes
3	3	yes
3	8	yes

3	3	yes
3	3	yes
3	3	yes
3	8	yes
3	3	yes
3	8	yes
3	3	yes
4	4	yes
4	4	yes
4	4	yes
4	4	yes
4	4	yes
4	4	yes
4	4	yes
4	4	yes
4	4	yes
4	4	yes
4	4	yes
5	5	yes
5	5	yes
5	1	yes
5	5	yes
5	5	yes
5	5	yes
5	1	yes
5	5	yes
5	5	yes
5	5	yes
6	6	yes
6	6	yes
6	6	yes
6	6	yes
6	6	yes
6	6	yes
6	6	yes
6	6	yes
6	6	yes
6	6	yes
7	7	yes
7	7	yes
7	7	yes
7	7	yes
7	7	yes
7	9	no
7	7	yes

7	7	yes
7	7	yes
7	7	yes
8	8	yes
8	8	yes
8	8	yes
8	8	yes
8	8	yes
8	8	yes
8	8	yes
8	8	yes
8	8	yes
8	8	yes
9	9	yes
9	9	yes
9	9	yes
9	9	yes
9	9	yes
9	9	yes
9	9	yes
9	9	yes
9	9	yes

Table 7. Digit recognizer tests.