Charles University in Prague

Faculty of Mathematics and Physics

BACHELOR THESIS

Juraj Citorík

Hlasové ovládání pro efektivní editaci textu

Department of Software Engineering

Supervisor of the bachelor thesis:	RNDr. Jakub Lokoč, Ph.D.
Study programme:	Computer Science
Curriculum:	General Computer Science

Prague 2013

None of this would be possible without the support of my family. Thank you.
I would also like to thank my thesis supervisor, RNDr. Jakub Lokoč, Ph.D. for helpful advice, patience and trust. I would like to extend my sincere gratitude to those who went above and beyond to help me, including Martina Černíková, Lukáš Ďurovský, Zuzana Abelovská, Dominik Meňhart, Martin Klepáč, Tomáš Susedik, Michal Bilanský, Samuel Bartoš and Vendula Michlíková.

Názov práce: Hlasové ovládání pro efektivní editaci textu

Autor: Juraj Citorík

Katedra: Katedra softwarového inženýrství

Vedúci bakalárskej práce: RNDr. Jakub Lokoč, Ph.D., Katedra softwarového inženýrství

Abstrakt: Cieľom tejto práce je poskytnúť úvod do problematiky digitálneho spracovania zvuku a rozpoznávania reči. V texte je popísaných niekoľko vybraných deskriptorov reči a algoritmov spojených s problematikou. Tieto sú použité v implementácii jednoduchého hlasom ovládaného textového editoru a .NET knižnice. Deskriptory sú porovnané s ohľadom na rýchlosť a presnosť pri použití v systéme rozpoznávania príkazov pre textový editor a to v systéme závislom alebo nezávislom na hovoriacom. Knižnica tried poskytuje jednoduchý spôsob implementácie hlasového ovládania závislého na hovoriacom v obmedzenej doméne príkazov v ľubovoľnom programe. Editor textu umožňuje užívateľovi priradiť hlasové povely k zabudovaným funkciám programu, čo napríklad umožňuje aj neskúseným užívateľom používať pokročilé funkcie bez nutnosti predošlého učenia sa napríklad klávesových skratiek. Tento prístup je navyše nezávislý na jazyku a je použiteľný aj pre ľudí s poruchami reči, čo momentálne rozšírené riešenia neumožňujú. Výsledky experimentov ukazujú, že prezentované deskriptory a algoritmy sú, za predpokladu dostatočnej kvality nahrávky, dostatočne efektívne pre použitie pri rozpoznávaní príkazov v systéme závislom na hovoriacom.

Kľúčové slová: hlasové ovládanie, digitálne spracovanie zvuku

Title: Voice control for effective text editing

Author: Juraj Citorík

Department: Department of Software Engineering

Supervisor: RNDr. Jakub Lokoč, Ph.D., Department of Software Engineering

Abstract: The aim of this thesis is to provide a comprehensive introduction to digital sound processing and speech recognition. Selected speech recognition features as well as algorithms are introduced and utilized in a voice controlled text editor and a .NET class library. The performance of the features is evaluated in both speaker-dependent and speaker-independent recognition of commands related to text editing. The library provides a straightforward way of implementing a speaker-dependent, domain-constrained voice recognition in an arbitrary application. It is used in a simple voice controlled text editor. The editor allows the user to assign voice commands to built-in actions. In this way, it is possible for inexperienced users to access and use advanced features of the program without having to learn complex workflows. Moreover, this approach is language-agnostic and can even be used by people with speech impairments as opposed to majority of presently used voice recognition systems. The results of the experiments indicate that, given a recording of sufficient quality, the presented features and algorithms provide an effective means to implement a speaker-dependent speech recognition system, which can be used in a voice controlled text editor.

Keywords: voice recognition, digital sound processing

Contents

Introduction

Speech has always been the dominant means of human communication. This preference for spoken language communication, however, hasn't yet been reflected in the way humans interact with computers. Most computers utilize a graphical user interface, which depends on keyboard input and mouse clicks. Nowadays, consumers place a premium on simplicity of use and therefore it is essential to create human-computer interfaces that allow for a more natural interaction, gentle learning curve and thus higher productivity. One of the means to achieve this is speech recognition.

In the following chapters we will explore fundamentals of digital sound processing and speech recognition. This includes basic properties of sound and human speech, digitization of sound, speech recognition features and algorithms. These will be used as a foundation to build a voice controlled text editor and a .NET class library. Finally, the performance of the various speech recognition features will be assessed in both speaker-dependent and speaker-independent system for voice control with focus on commands related to text editing.

Motivation

Novice users and expert users alike can benefit from a spoken language interface. It is frequently difficult for novice users to control a new program, where simple actions require the knowledge of the program's interface conventions and involve manipulating several windows, checkboxes and sliders. This is in stark contrast to the simplicity of merely saying what the user wants to do. Moreover, professional users can use voice commands to avoid unnecessary obstacles in their workflow. For instance, a graphic designer might invoke a text formatting command using his or her voice while using the mouse to pinpoint the area to which the desired action should be applied. This results in a more fluid workflow and thus increased productivity.

Speech recognition features

Speech recognition features describe the speech signal in a way that allows us to find for a spoken command a matching command already present in the voice command database. In the following chapters, we will explore various speech recogniton features, such as *Mel Frequency Cepstral Coefficients*, as well as speech processing algorithms and techniques that allow us to create a voice-controlled text editor.

We will evaluate the performance of the features in one of the following chapters. The criteria that are of interest to us are speed of extraction and accuracy.

Voice control for text editing

The set of actions used in a typical text editing task is limited which allows us to provide the user with a list of available commands and prompt him or her to record the corresponding phrase. In this way, a local speaker-dependent voice command database is created.

This database is then utilized when the user uses the voice control to determine the correct action with respect to the spoken utterance. The fact that the user creates his or her own command database renders it usable for people with mild speech impairments and people who speak with a strong accent. It is also language-agnostic.

Speaker-independent speech recognition

We will also examine the perfomance of a speaker-independent voice recognition system, which will use a database of text editing commands recorded by several users. We will assess the accuracy of this database, as well as its perfomance when different underlying speech recognition features are used.

Chapters overview

Chapter 1 contains introduction to digital sound processing and speech recognition. It also provides descriptions of three speech recognition features: *Mel Frequency Cepstral Coefficients*, *Linear Predictive Coding* and *Perceptual Linear Prediction*. Algorithms for tackling voice activity detection and command matching are presented as well.
Chapter 2 discusses related works and techniques used in state-of-the-art speech recognition.
Chapter 3 includes details about the implementation of the .NET voice control library.
Chapter 4 contains the specification of the voice controlled text editor. A set of commands to be used in the program is determined. This chapter also specifies problems that have to be tackled so as to implement a functional solution.
Chapter 5 specifies the methodology of feature performance testing and discusses the results of the tests.

1. Speech and Sound Processing Fundamentals

1.1 Sound

Sound is a wave of pressure caused by an oscillating object that propagates through a medium, such as air. The wave includes zones where air molecules are in a compressed configuration and zones where air molecules are less compressed. The former zones are called *compressions* and the latter zones are called *rarefactions*. The alternating configurations of compression and rarefaction can be depicted by a graph of a sine wave as shown in Figure 1.1

We often use the term *signal* or *audio signal* to refer to a sound wave. A signal is a function that conveys information about the behaviour or attributes of some phenomenon. Throughout the remainder of this text we will use terms "speech" and "speech signal" interchangeably.

1.1.1 Properties of Sound

The sound *wavelength* λ is the distance between two subsequent maximal compressions or minimal rarefactions as depicted in Figure 1.1.
The *frequency* f of a sound is defined as:

$$f = \frac{c}{\lambda}$$

Where c is the speed of sound in the corresponding environment.

Signal *amplitude* is the amount by which the signal differs from zero. Signal *magnitude* is defined as the absolute value of amplitude. Signal *power* is defined as magnitude squared.

In practice, due to the vast range of sound pressure levels the human ear can detect, it is convenient to use a logarithmic *decibel scale* for sound intensity.

The sound's pressure level (SPL) in decibels (dB) is actually a comparison of its pressure level P to the reference pressure level P_0 that is equal to 0 dB and

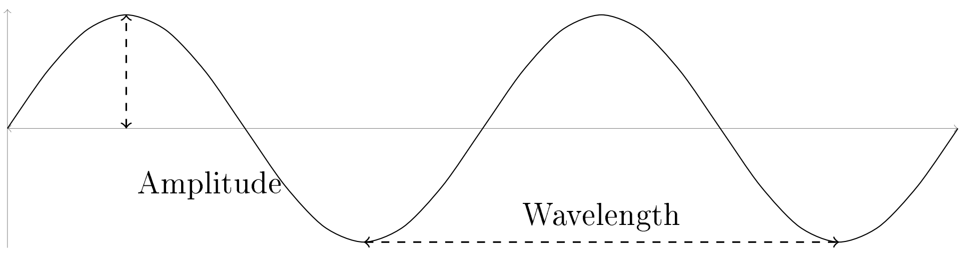


Figure 1.1: The sinewave can be used to describe a sound wave. The peaks of the sine curve correspond to maximal compression, while the valleys indicate minimal rarefaction. Two important properties of a sound wave, amplitude and wavelength, are shown in the figure as well.

Sound	**dB Level**	**Times higher than TOH**
Threshold of hearing	0	10^0
Light whisper	10	10^1
Quiet conversation	40	10^4
Normal conversation	60	10^6
Heavy truck traffic	90	10^9
Pain threshold of ear	120	10^{12}
Sonic boom	140	10^{14}
Rocket engine	180	10^{18}

Table 1.1: Intensity and decibel levels of various sounds.[1]

corresponds to the threshold of human hearing (TOH):

$$SPL\,(dB) = 20\log_{10}\left(\frac{P}{P_0}\right)$$

Decibel levels of selected sounds are shown in Table 1.1.

1.2 Digital Signal Processing Fundamentals

1.2.1 Sound Digitization Process

To be able to work with sound signals on a computer, we have to digitize them. In this process, a continuous analog signal is turned into a discrete signal by a process called *sampling* (Figure 1.2). This process is carried out by a analog-to-digital converter (ADC), which samples the continuous signal at regular intervals and stores the amplitude corresponding to a particular moment in a *sample.* The *sample rate* is the number of times an analog signal is measured (sampled) per second. The conversion of the amplitude of each sample to a binary number is called *quantization.* The number of bits used for quantization is referred to as *bit depth.* Bit depth and sample rate (sampling frequency) are the two most important factors that determine the quality of a digital audio system[5].

A digital signal can only contain a limited range of frequency components. The limit is given by the sampling frequency as stated in the *Nyquist Sampling Theorem*:

Theorem 1. *For a signal sampled at a rate R the highest frequency that the signal can contain is $R/2$. Otherwise distortion appears in the digitized signal.*

1.2.2 Waveform and Frequency Spectrum

With the exception of pure sine waves, sounds are made up of many different frequency components vibrating at the same time. The particular characteristics of a sound are the result of the unique combination of frequencies it contains. Sounds contain energy in different frequency ranges, or *frequency bands.* Once we obtain the frequency components of a signal, we can represent the signal in the frequency domain with a spectrogram, as shown in Figure 1.3, which also displays the *waveform* of the signal, representing the amplitude of the signal in the time domain.

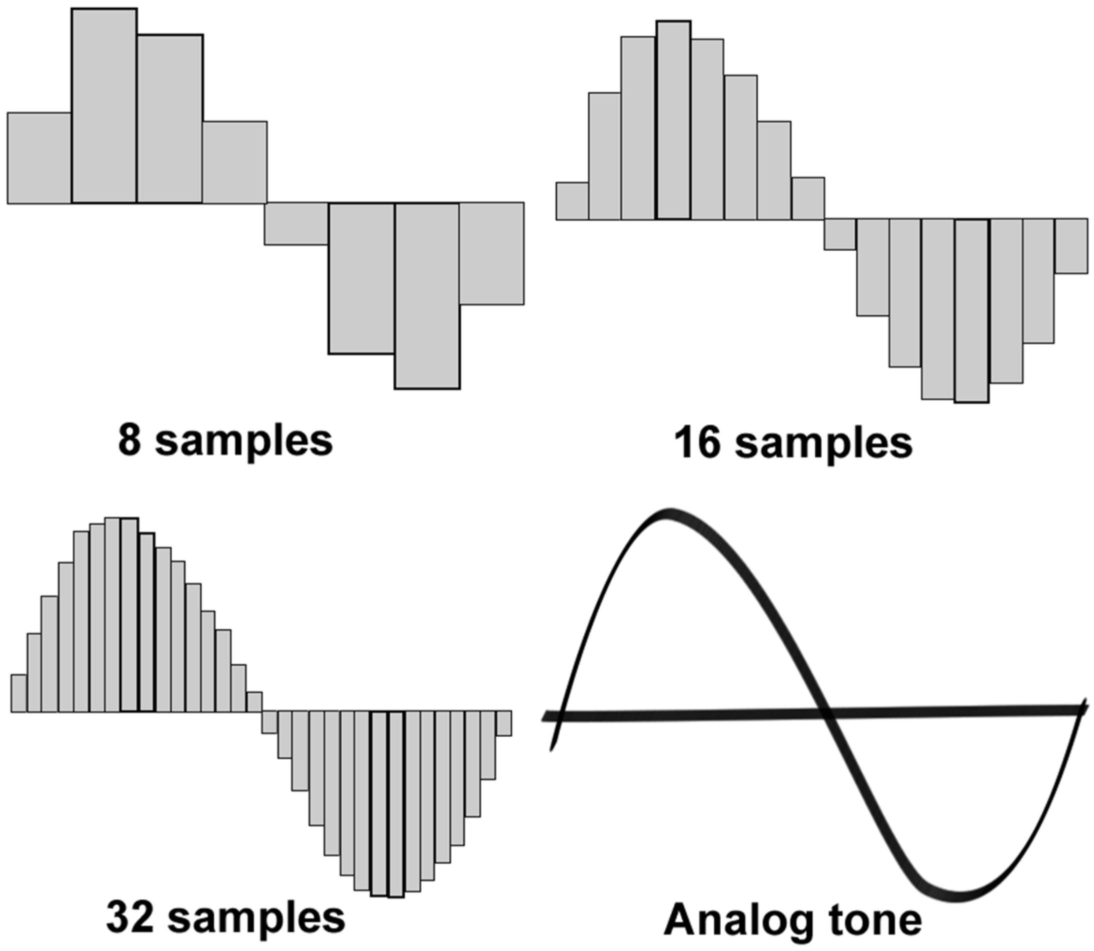


Figure 1.2: Digitization of a simple sound signal with various sample rates[7]

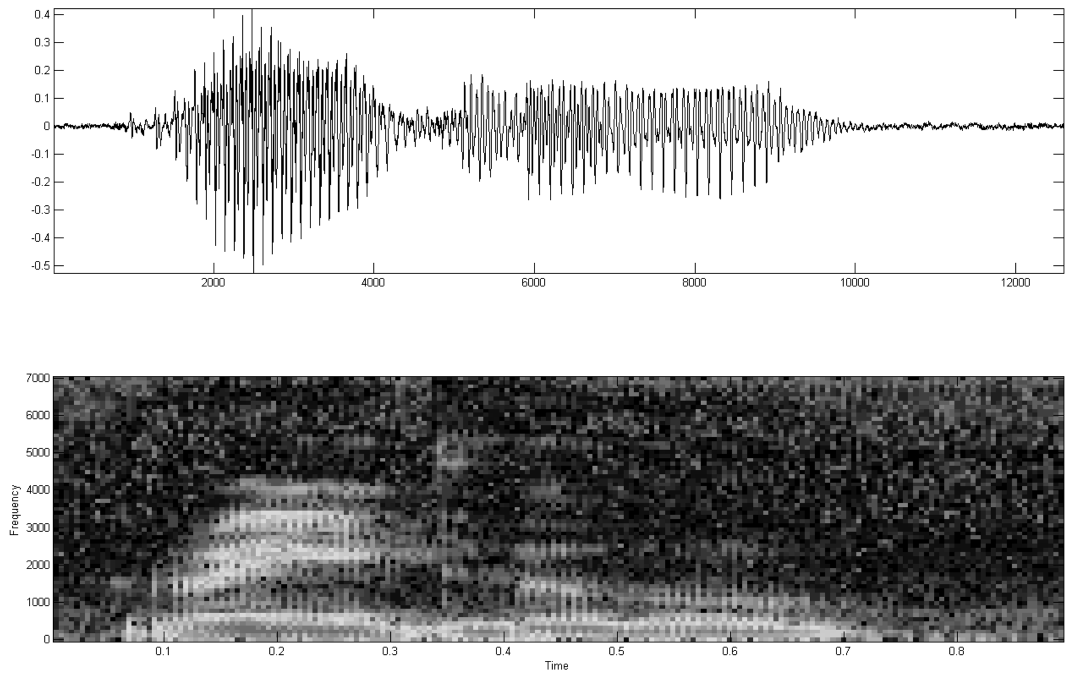


Figure 1.3: Speech signal waveform, and its spectrogram. The light spots indicate high intensity of the frequency components at the corresponding time.

1.2.3 The Discrete Fourier Transform

To obtain the frequency spectrum of a signal, the *Fast Fourier Transform* (FFT) is used, which is an efficient implementation of the Discrete Fourier Transform (DFT). The DFT of a signal $x_N[k]$ is defined as

$$X_N[k] = \sum_{n=0}^{N-1} x_N[n] e^{-j2\pi nk/N} \quad 0 \leq k < N$$

And since

$$e^{-j2\pi nk/N} = \cos\left(2\pi nk/N\right) - j\sin\left(2\pi nk/N\right)$$

we can intuitively see that DFT is a decomposition of a signal into a linear combination of sines and cosines of various frequencies. Figure 1.4 shows the FFT of the signal from Figure 1.3 which is sampled at 14 kHz . Note that it contains information about frequencies up to 7 kHz, which corresponds to the *Nyquist frequency* of a signal sampled at 14 kHz. The FFT of a simple signal is shown in Figure 1.5.

Straightforward implementation of the above formula yields a computational complexity of $\mathrm{O}\big(n^2\big)$, but if the FFT algorithm is used to compute the transform, the complexity is $\mathrm{O}\big(n \ \log n\big)$. Detailed description of the FFT algorithm can be found in [1]

The result of an N-point transform is a vector containing N *frequency bins*. Each bin contains intensity information about a range of frequencies. The range is determined by N, the sampling rate of the signal and the index of the bin. For instance, the frequency band corresponding to a bin with a zero-based index k, in a transform of size N of a signal sampled at fs Hz is:

$$k\left(\frac{fs}{N}\right) \text{ to } (k+1)\left(\frac{fs}{N}\right) \text{ Hz}$$

The Inverse Discrete Fourier Transform

To transform a signal from the frequency domain back to the time domain, the Inverse Discrete Fourier Transform (IDFT) is used:

$$x_N[n] = \frac{1}{N}\sum_{k=0}^{N-1} X_N[k] e^{-j2\pi nk/N} \quad 0 \leq n < N$$

The Discrete Cosine Transform

Closely related to the Discrete Fourier Transform and frequently used in speech recognition is the Discrete Cosine Transform (DCT). The DCT of a real-valued signal $x_N[k]$ is defined as

$$C[k] = \sum_{n=0}^{N-1} x[n] \cos\left(\pi k \left(n + 1/2\right)/N\right) \quad 0 \leq k < N$$

Compared to DFT, it is often the case that the coefficients of DCT are concentrated at lower indices, which means we can approximate the signal with fewer coefficients.[1]

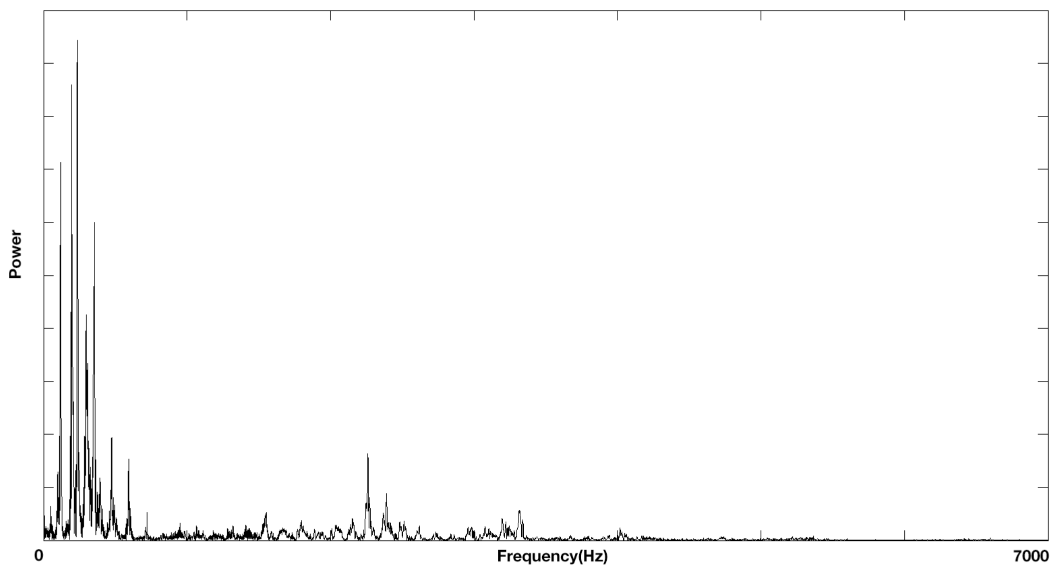


Figure 1.4: Fourier transform transform of the signal from Figure 1.3

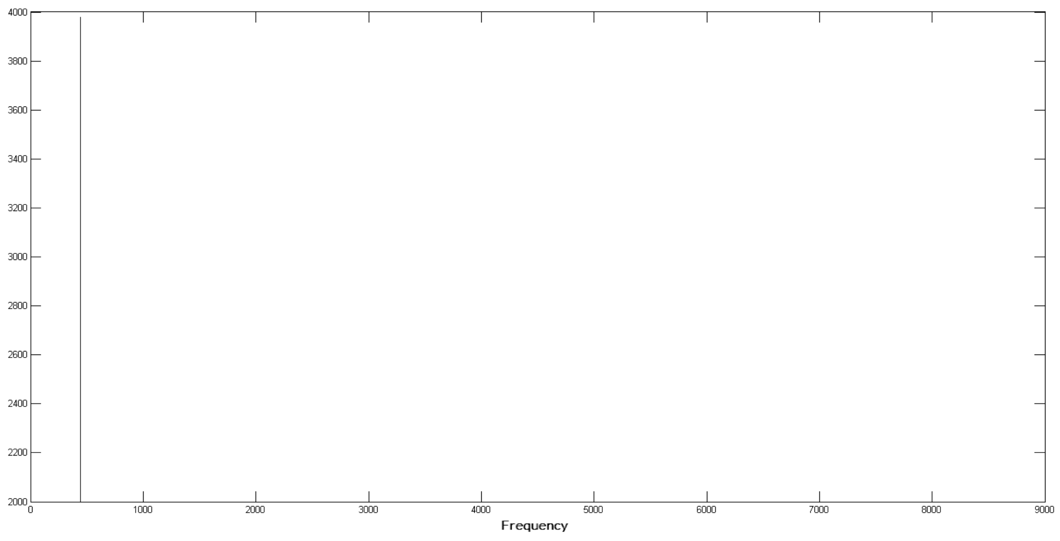


Figure 1.5: Fourier transform of a 440 Hz sine wave

1.3 Windowing

In digital signal processing (DSP), a signal is rarely processed in one piece. The signal is usually divided into *frames* and the processing is applied to individual frames. However, a problem called *spectral leakage* may occur, which causes an FFT bin to contain energy from adjacent frequency bins. To counter this, we use *windows*.

1.3.1 Spectral Leakage

The FFT algorithm assumes that the input signal is periodic throughout all time and that the period length is the same as the length of the input. If the input contains a non-integral number of cycles, spectral leakage occurs. The problem is illustrated in figure 1.6, which shows the spectrum of a frame taken from the signal whose fourier transform (computed for the whole signal at once) is shown in figure 1.5. It is clear that the information about frequency components is inaccurate. One way to amend this is to divide the signal into frames that only contain integral number of cycles. In many cases, however, this is either not possible or a constant frame size is required.

Figure 1.6 shows the spectra obtained from a frame containing non-integral number of cycles, using three types of windows. Spectral leakage is clearly present.

1.3.2 Window Characteristics

Figure 1.7 shows characteristics that describe a window and tell us how it will perform. The center of the main lobe occurs at each frequency bin of the signal. The width of the main lobe is given in bins and determines the frequency resolution of the window. However, as the main lobe narrows, its energy is distributed into its side lobes which decreases the accuracy of the information about amplitude.

1.3.3 Windows

Applying a window is equivalent to convolving the window with the input. Even when no window is used, the input is in fact convolved with a rectangular window of uniform height, therefore no window is sometimes called Rectangular or Uniform window. The most frequently used windows in speech recognition are rectangular, Hann and Hamming window.

Rectangular (Uniform) Window

An N-sample Rectangular Window is defined by:

$$w(n) = 1$$

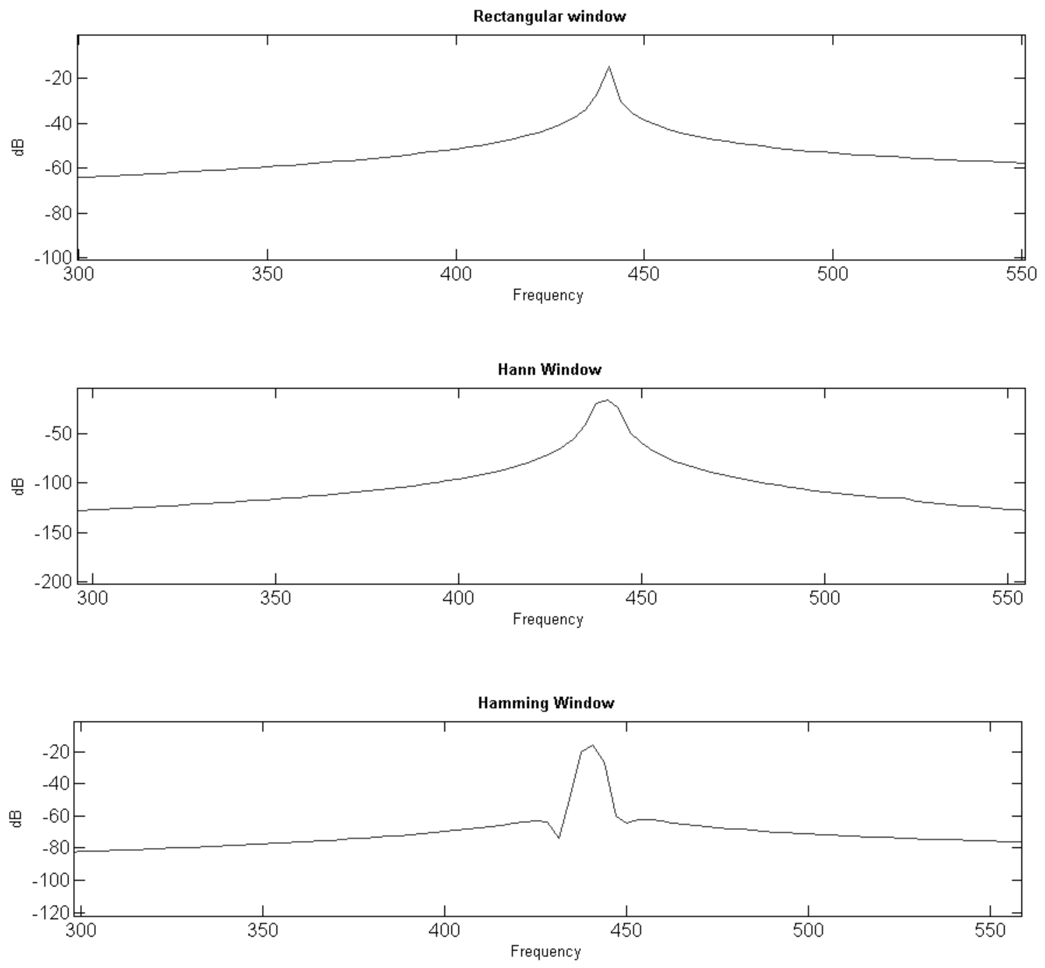


Figure 1.6: Spectra obtained from a part of a signal containing a 440 Hz sine wave, using no window (also called Rectangular or Uniform window), Hann window and Hamming window.

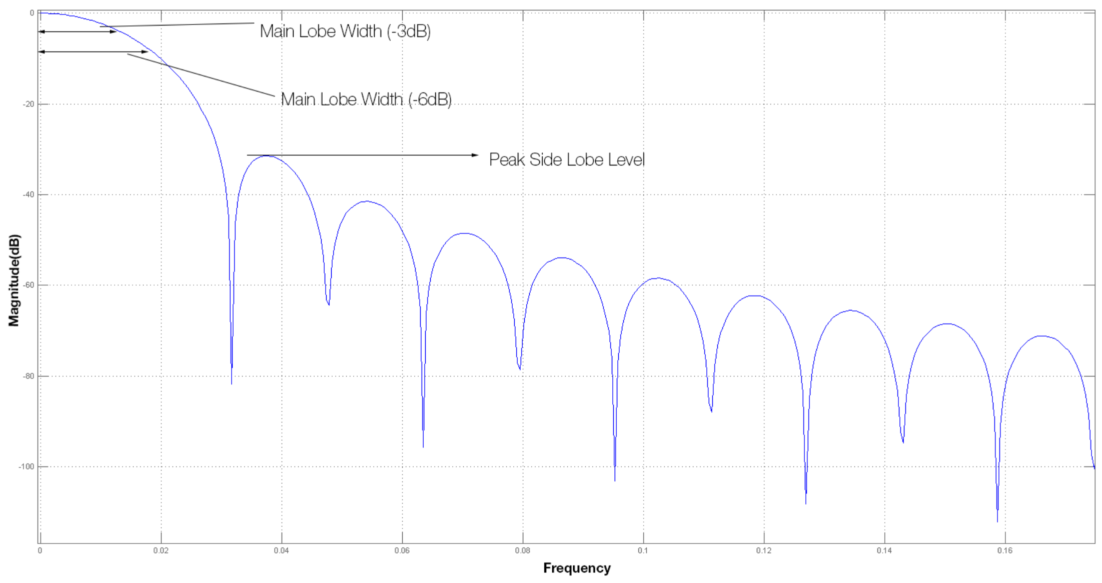
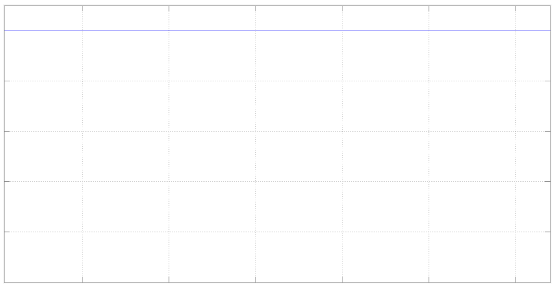


Figure 1.7: Properties of a window function


(a) Rectangular window plot

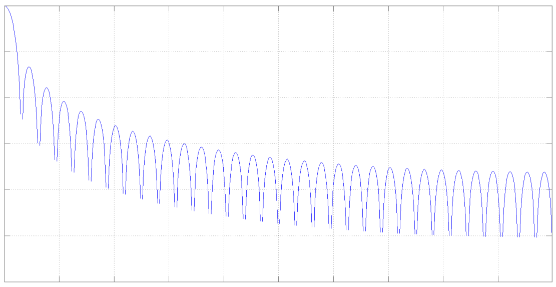
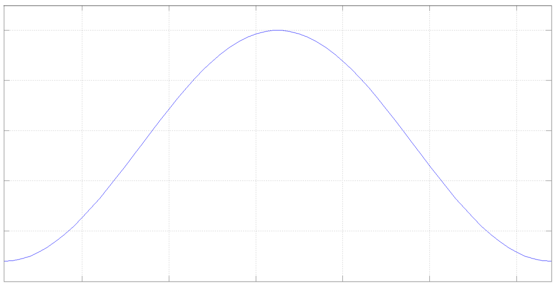

(b) Rectangular window magnitude response

Figure 1.8: Rectangular window plot and magnitude response


(a) Hamming window plot

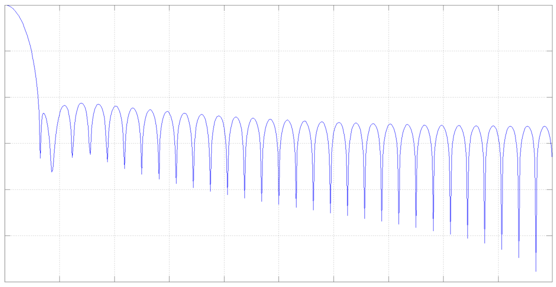

(b) Hamming window magnitude response

Figure 1.9: Hamming window plot and magnitude response

Hann Window

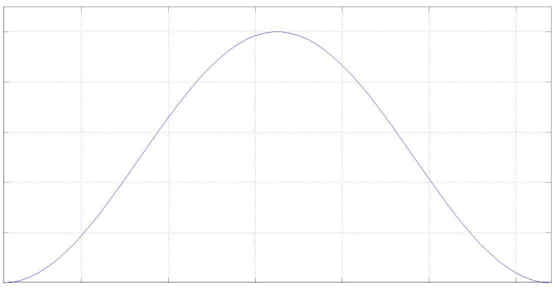
An N-sample Hann Window is defined by:

$$w\left(n\right) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right)$$

Hamming Window

An N-sample Hamming Window is defined by:

$$w\left(n\right) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)$$


(a) Hann window plot

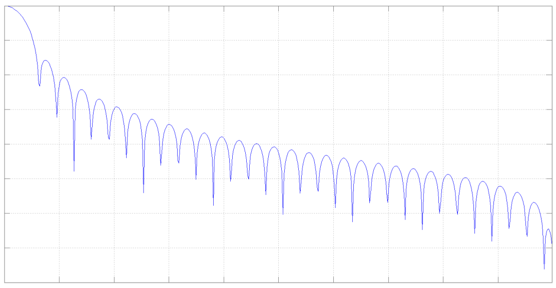

(b) Hann window magnitude response

Figure 1.10: Hamming window plot and magnitude response

Window	-3 dB Main Lobe Width (bins)	-3 dB Main Lobe Width (bins)	Maximum Side Lobe Level (dB)
Rectangular	0.89	1.21	-13
Hann	1.44	2.00	-31
Hamming	1.30	1.82	-43

Table 1.2: Properties of the Rectangular, Hann and Hamming window. Rectangular window has great frequency resolution due to a narrow main lobe. On the other hand, the Hann and Hamming windows have better amplitude resolution thanks to their low maximum side lobe level.

An overview of the properties of the aforementioned window functions can be found in Table 1.2. Figures 1.8,1.9 and 1.10 display plots and magnitude response of these windows.

1.4 Speech

1.4.1 Production of Speech

When humans produce speech, the air flows first through the glottis, which is the combination of the vocal cords and the space between them, then through the throat and finally, mouth. There are three ways a speech signal can be produced:

- **voiced excitation** A periodic sequence of pulses (pulse train) is generated by air pressure that forces the *closed glottis* to open. The pulse train determines the "*fundamental frequency*" and in the case of human speech it usually lies between 80 Hz to 350 Hz. (Figure 1.11)
- **unvoiced excitation** The air passes through the open glottis and through a narrow passage in the throat or mouth. This creates a noise signal. (Figure 1.12)
- **transient excitation** The air pressure is raised by a closure in the throat or mouth and then a sudden opening of the closure causes the pressure to drop immediately. (Figure 1.13)

The spectrum of a speech signal is determined by the shape of the vocal tract(throat,teeth,tongue and lips).

1.4.2 Speech Signal Characteristics

There are a few properties of the speech signal worth noting:

- All the information necessary to understand human speech is contained within 0–4 kHz
- The fundamental frequency ranges from 80 Hz for a large man to 350 Hz for a child or a woman

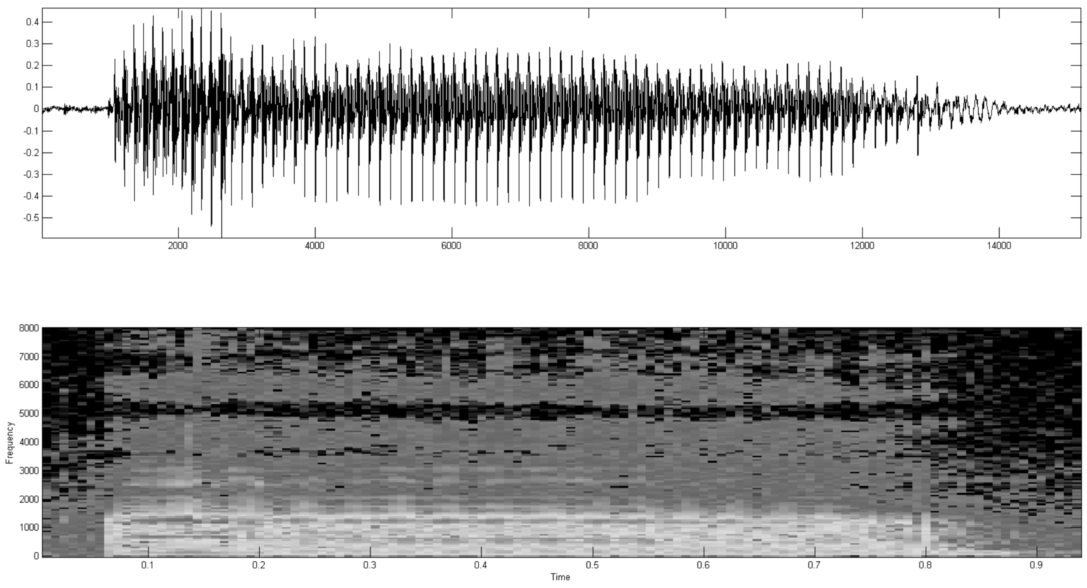


Figure 1.11: Waveform and spectrum of a speech sound produced by voiced excitation

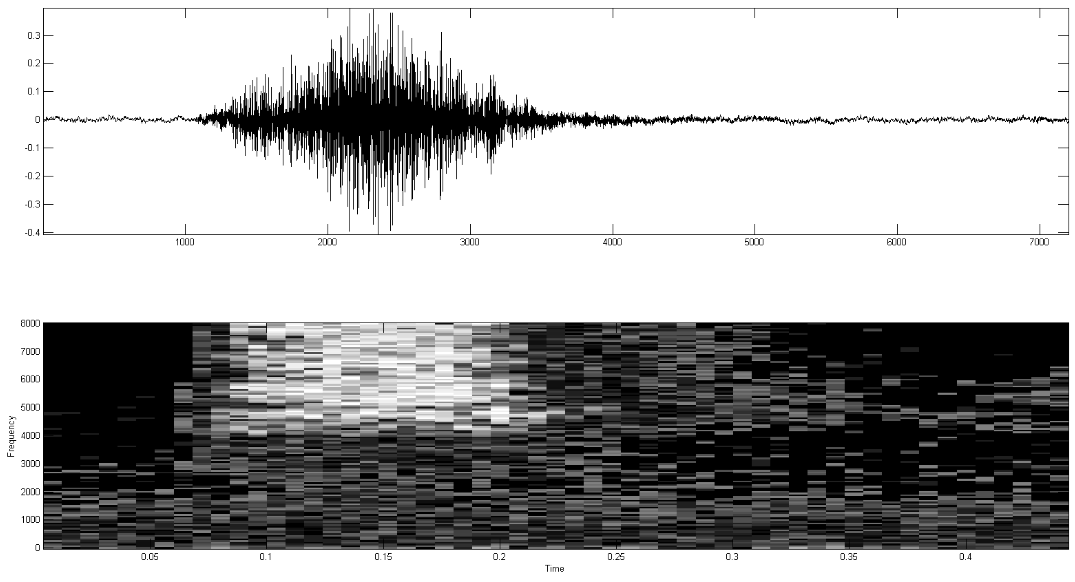


Figure 1.12: Waveform and spectrum of a speech sound produced by unvoiced excitation

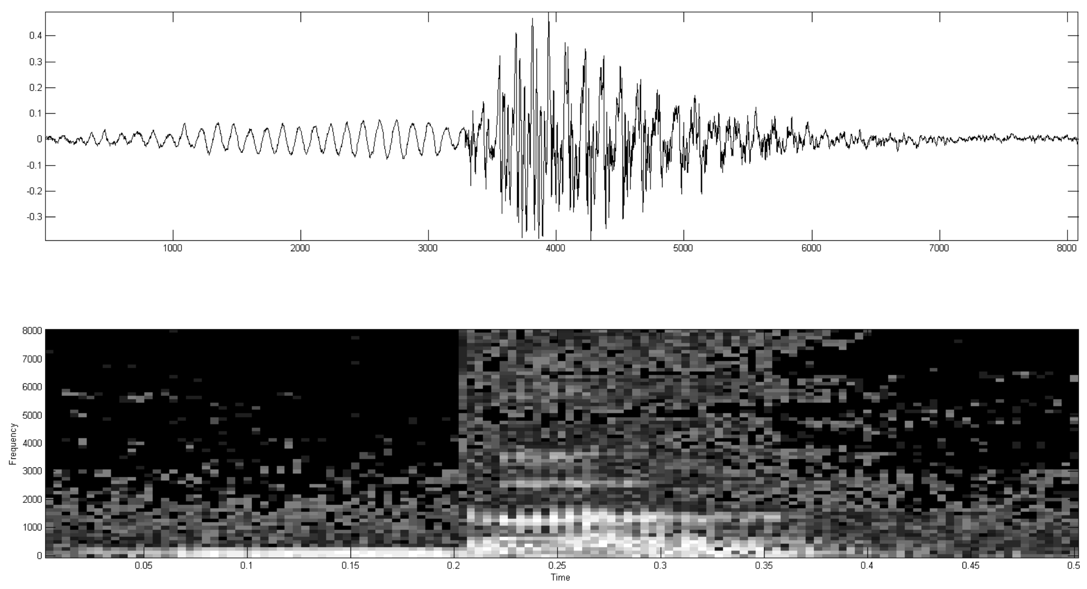


Figure 1.13: Waveform and spectrum of a speech sound produced by transient excitation

- Upon closer examination of a speech signal spectrum, on may notice peaks in the spectrum at *formant frequencies*

$$(2n-1) * 500\text{Hz} \quad n = 1, 2, 3, \ldots$$

1.4.3 Speech Perception

The speech understanding process works as follows. The signal is passed to a part of the inner ear called cochlea1.14. The cochlea can be roughly viewed as a filter bank (a set of filters which extract a certain frequency band from the signal and attenuate the frequencies outside the band). The spectrum from the cochlea is converted by neural transduction into signals for the auditory nerve, which *roughly correspond to a speech feature extraction component.* It is currently not known how the signal from the auditory nerve is mapped into the language system and how comprehension is achieved in the brain[4].

Mel Frequency Scale

Researchers have found that the perception of sounds at different frequencies is not linear in nature. In hope that a more natural model of human ear sensitivity is achieved, the mel frequency scale has been created. It is linear below 1 kHz and logarithmic above. The mel frequency f_{mel} for frequency f can be approximated as

$$f_{mel} = 1125 \ln(1 + (f/700))$$

The motivation behind this is the fact that the passbands(the range of frequencies that are not attenuated by the filter) as well as the central frequencies of the filters in the cochlea are non-linearly spaced on the frequency scale. If, however, we map

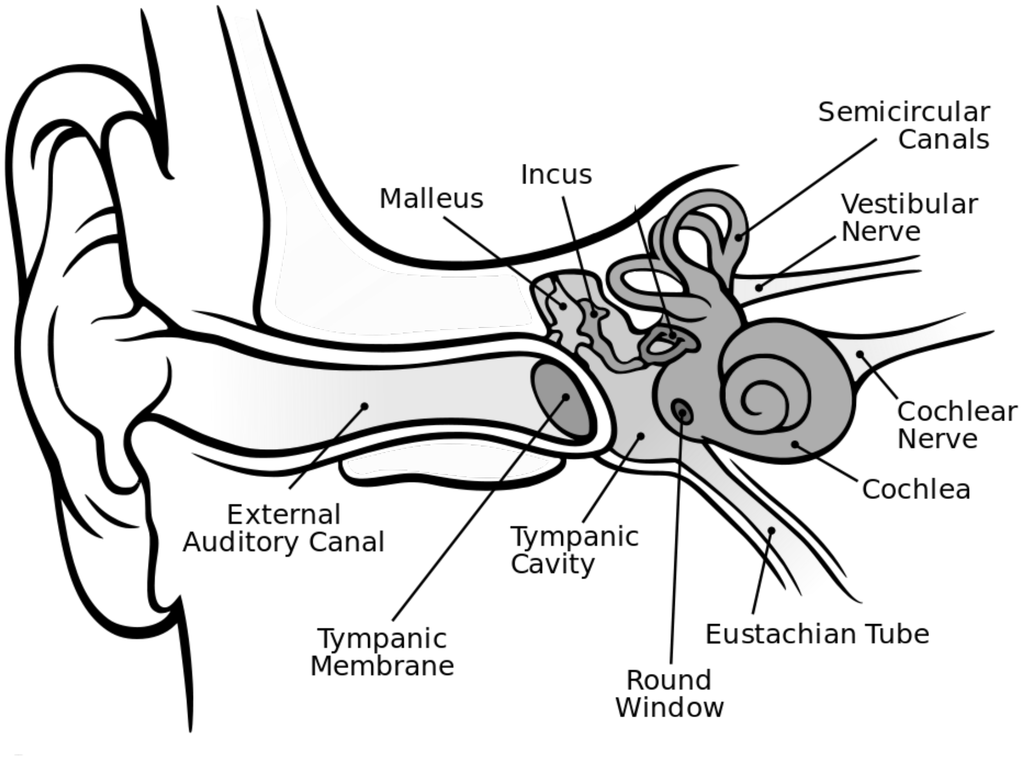


Figure 1.14: Human ear anatomy[6]

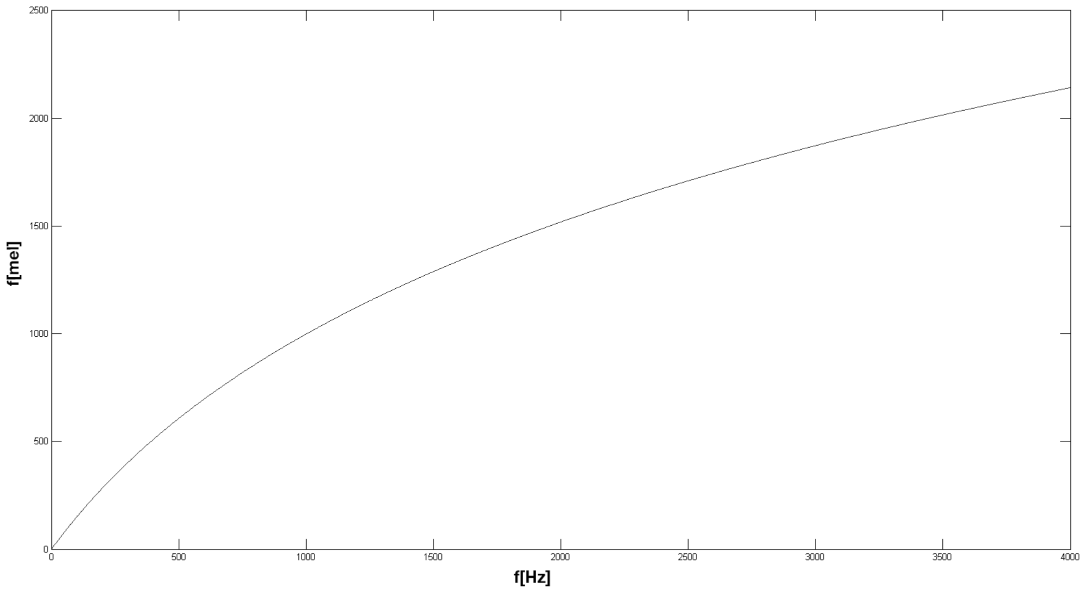


Figure 1.15: The mel frequency scale

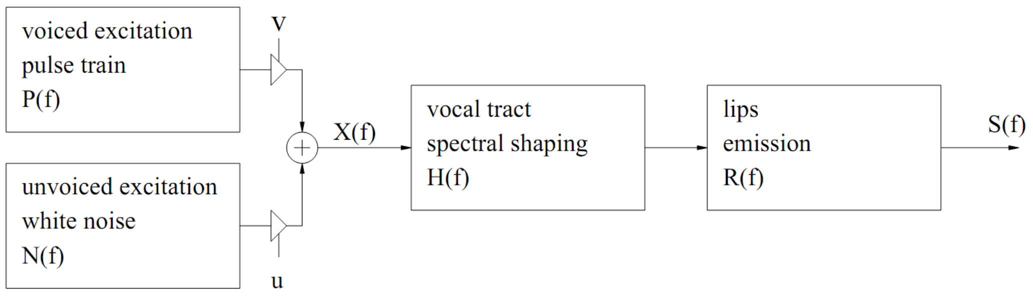


Figure 1.16: A simple speech production model[3]

the frequency scale to the mel scale, the passbands and their central frequencies are spaced linearly on this scale. Figure 1.15 depicts the mel scale.

When we describe a speech signal, we want to approximate the way the speech is perceived by the human ear. Thus we examine the energy contained in frequency bands that are the same as the bands that the cochlea extracts from the sound. However, the bands are not linearly spaced on the frequency scale. If we use the mel scale to find the frequency ranges to examine, we get more accurate results.

1.4.4 A Simple Speech Production Model

Using the information presented so far about speech production, we can now create a simple model(Figure1.16) for speech production that will help us describe speech recognition features later in this chapter. In this model, voiced excitation is modeled by a pulse generator v with spectrum given by P(f). Unvoiced excitation is modeled by a noise generator u with spectrum N(f). Amplitude of u and v can be adjusted. The next box models spectral shaping cause by varying shape of the vocal tract. This section of the model is of great interest to us, as we will later see, due to the fact that for a speech recognizer, the most valuable information is contained in the way the spectral shape of the speech signal changes in time. This shaping is done by H(f). Finally, the lips characteristics are applied to the signal by R(f) to give the spectrum of the speech signal S(f)[3]

$$S\left(f\right) = \left(vP\left(f\right) + uN\left(f\right)\right)H\left(f\right)R\left(f\right) = X\left(f\right)H\left(f\right)R\left(f\right)$$

1.5 Speech Recognition Features

Audio features are a means of representing a signal in a way that reflects the most important properties for our application. In case of speech recognition, there are several features that provide a representation of a speech signal that is close to the way humans perceive speech. In this section, we will examine three of them.

1.5.1 Mel Frequency Cepstral Coefficients

Mel Frequency Cepstral Coefficients(MFCC) have been the dominant features used in speech recognition. The extraction of MFCCs consists of several steps,

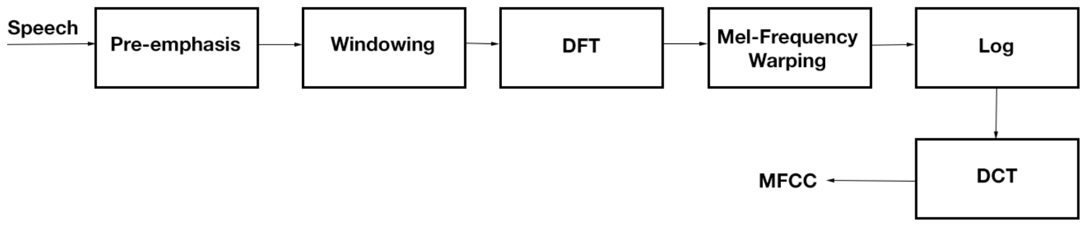


Figure 1.17: The MFCC extraction process

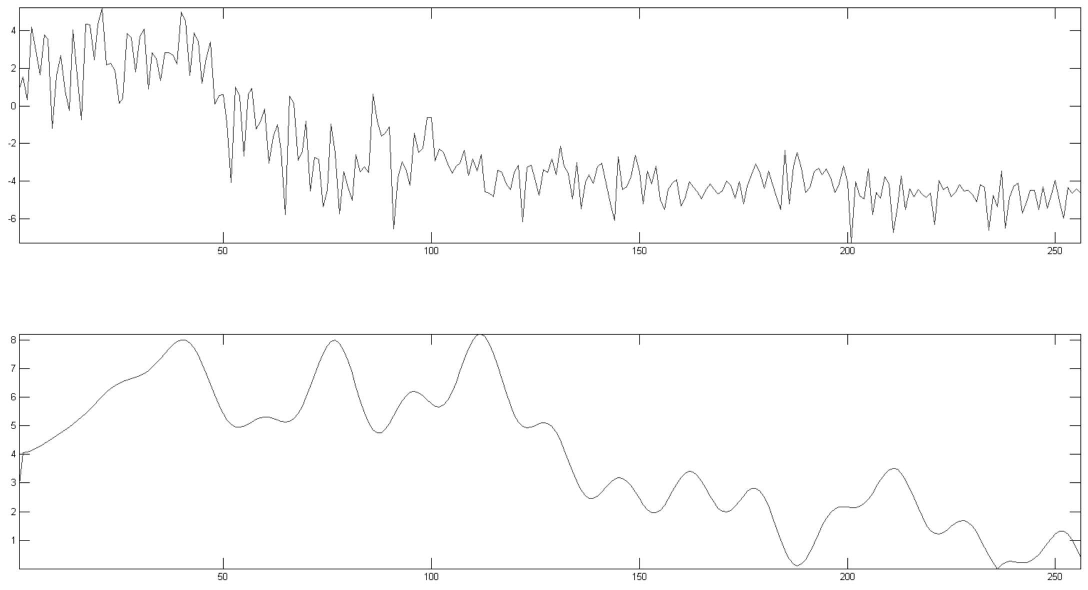


Figure 1.18: From top: The spectrum of a speech signal and a smoothed version of that signal

as shown in Figure1.17. To understand the intuition behind MFCC, we use the speech production model from 1.4.4. The goal of MFCC is to obtain a smooth spectrum from a speech signal. Both an original and a smoothed spectrum are shown in Figure1.18. The ripples in the spectrum initially obtained from the speech signal are caused by the excitation part of our model and MFCCs attempt to remove them. In this way, the influence of the voice pitch is eliminated, in other words it doesn't matter if the speech is produced by a man, a woman or a child.

Pre-emphasis Filtering

The input signal is first passed through a pre-emphasis filter, which amplifies higher frequencies. This accounts for -6 dB magnitude decrease per octave for higher frequencies in a typical speech signal[3]. The filter application to a signal s to obtain s_p is denoted as:

$$s_p(k) = s(k) - 0.97 s(k-1)$$

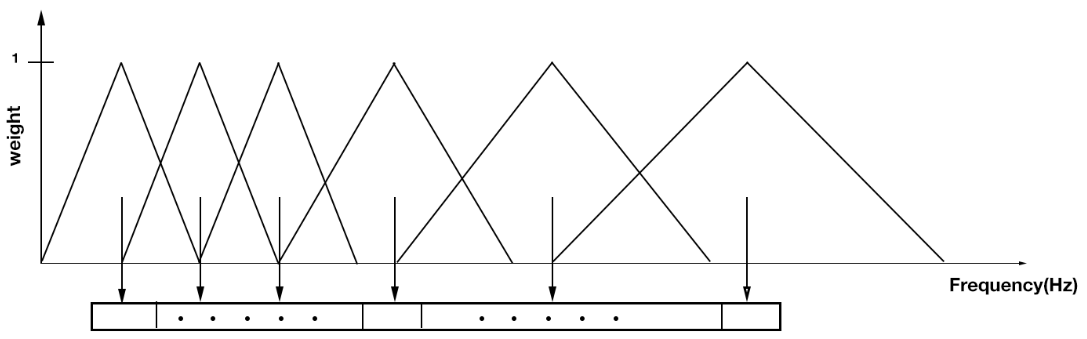


Figure 1.19: A triangular filter bank. The weight of the bin in a group sum is determined by a triangular function. Every triangle corresponds to one frequency band

Framing and Windowing

After applying the pre-emphasis filter, the signal is divided into frames, to which a *Hamming window* is applied to diminish spectral leakage. The size of the frames is usually between 16 ms and 40 ms and overlap is set between 6 ms to 30 ms[3].

DFT

In this step, DFT is computed for each frame, which gives us the rippled spectrum of the frame.

Mel-Frequency Warping

To reflect the nonlinear perception of frequencies, we group the bins from the DFT into bands that are linearly spaced on the mel scale. The values of the frequency bins within a group are added together. This also reduces the dimensionality of the resulting features. The number of groups is usually set to 20.

In some implementations, the sum of the frequency bins within a group is a weighted sum. To determine the weights, a set of triangular functions is used. Since we are operating in the frequency domain, we call this set a *triangular filter bank*. The peaks of the triangles are spaced equally on the mel scale and every triangle starts at the center frequency of the previous triangle. An example of a (smaller) triangular filter bank is shown in Figure1.19. To compute a weighted sum for a band of N frequency bins, we sample the corresponding triangular function N times to obtain weights for the bins. We then calculate the weighted sum using these weights.

Logarithm of the Frequency Spectrum

For the equation in 1.4.4, it holds that:

$$\begin{aligned}
\log\left(|S(f)|^2\right) &= \log\left(|H(f)|^2 \cdot |U(f)|^2\right) \\
&= \log\left(|H(f)|^2\right) + \log\left(|U(f)|^2\right) \\
\text{where} \quad U(f) &= X(f) \cdot R(f)
\end{aligned}$$

If we knew $|U(f)|^2$, we could subtract it to get $|H(f)|^2)$. All we know, however, is that $|U(f)|^2)$ causes the ripples in the spectrum. If we looked at the spectrum as if it were a time signal, the ripples would be caused by a high frequency component. To eliminate this component, we can do a low-pass filtering, which attenuates high frequencies.

Discrete Cosine Transform

The DCT is applied to the log spectrum obtained in the previous step. The result of applying the DCT to the spectrum is called *cepstrum*.

Liftering

To remove the high frequency components (which cause the ripples in the spectrum), we set the coefficients with higher indices (these represent the higher frequencies) to zero. We are basically altering the spectrum of a spectrum (the cepstrum). This is called *liftering*. We are left with a number of non-zero coefficients, these are usually the first 14 (or fewer) coefficients[3]. These are the *Mel Frequency Cepstral Coefficients*

Delta Coefficients

In current state-of-the-art speech recognition systems, the first-order delta Δc_k and second-order delta $\Delta\Delta c_k$ are added to the MFCC features c_k[1].

$$\begin{aligned}\Delta c_k &= c_{k+2} - c_{k-2}\\ \Delta\Delta c_k &= \Delta c_{k+1} - \Delta c_{k-1}\end{aligned}$$

1.5.2 Linear Predictive Coding

Linear Predictive Coding (LPC) is another method of separating out the effects of excitation and vocal tract shaping from a speech signal. The technique models a signal as a linear combination of previous values. For instance, an approximation of a signal y at time n using p coefficients is denoted as:

$$y[n] = a[1]y[n-1] + a[2]y[n-2] + \ldots + a[p]y[n-p] + e[n]$$

Where p is called the LPC model order and $e[n]$ is the error of the prediction. The aim of LPC is to minimize the error, which can be expressed as the difference between the predicted signal value $\hat{y}[n]$ and the actual value $y[n]$:

$$e[n] = y[n] - \hat{y}[n] = y[n] - \sum_{k=1}^{p} a_k y[n-k]$$

or equivalently:

$$y[n] = \hat{y}[n] + e[n]$$

This formulation defines the signal as a weighted past samples and and an error signal. If we use the same system for speech production as for MFCCs, we can look at the error signal as the excitation signal and the predicted signal as the vocal tract filter responsible for the spectral shape of speech.

It is important to compute the predictor coefficients for a short speech segment since the speech properties change quickly over time. The predictors at time n are computed from a short segment of the signal around time n. We now define short-term speech and error segments to be used in equations for finding the predictors.

$$y_n[m] = y[n+m]$$
$$e_n[m] = e[n+m]$$

To minimize the error at time n, we are going to minimize the mean squared error at time n:

$$E_n = \sum_m e_n^2[m] = \sum_m \left(y_n[m] - \sum_{k=1}^{p} a_k y_n[m-k] \right)$$

Since we seek to minimize the error, we take the derivative with respect to a_i and equate it to 0, obtaining:

$$\sum_m \left(y_n[m] y_n[m-i] - y_n[m-i] \sum_{k=1}^{p} a_k y_n[m-k] \right) = 0$$

which can be written as:

$$\sum_m y_n[m] y_n[m-i] = \sum_{k=1}^{p} a_k \sum_m y_n[m-i] y_n[m-k]$$

and by setting

$$\phi_n(i,k) = \sum_m y_n[m-i] y_n[m-k]$$

can be expressed in a compact notation as:

$$\phi_n(i,0) = \sum_{k=1}^{p} a_k \phi_n(i,k) \quad 1 \le i \le p$$

which describes a set of p equations in p unknowns. Solving the system of equations gives us the predictor coefficients which can be used to describe a speech sound in a speech recognition system.

1.5.3 Perceptual Linear Prediction

The Perceptual Linear Prediction (PLP) utilizes the properties of human hearing to alter the speech spectrum. The resulting spectrum is then used in LPC analysis, as described in the previous section.

Spectral Analysis

The signal is divided into windows, usually 20 ms long[9], the Hamming Window is applied to each and the FFT is used to obtain the short-term spectrum of the speech signal. The square of the absolute value of the spectrum is taken to get squared power spectrum.

Critical-band Spectral Resolution

In this step, the squared power spectrum is warped into the *Bark scale*. Similar to Mel scale, Bark scale also attempts to warp the frequency scale so that it better reflects the way human ear perceives various frequencies. The scale ranges from 1 to 24, reflecting 24 critical bands of hearing[8]. The conversion from hertz to Bark is defined as:

$$b(f) = 13 \arctan(0.00076 f) + 3.5 \cdot \arctan\left((f/7500)^2\right)$$

To approximate the shape of the auditory filters in the cochlea, the warped spectrum is convolved with the power spectrum of the simulated critical band masking curve $\psi(\Omega)$. This is similar to the application of a triangular filter bank in the MFCC extraction process. The masking curve is given in [9] as:

$$\psi(\Omega) = \begin{cases} 0 & \text{for } \Omega < -1.3, \\ 10^{2.5(\Omega+0.5)} & \text{for } -1.3 \leq \Omega \leq -0.5, \\ 1 & \text{for } -0.5 < \Omega < 0.5, \\ 10^{-1.0(\Omega-0.5)} & \text{for } 0.5 \leq \Omega \leq 2.5, \\ 0 & \text{for } \Omega > 2.5. \end{cases}$$

The convolution results in a reduced resolution of the spectrum, which allows us to downsample. The sampling interval is chosen so that an integral number of spectral samples covers the whole band of frequencies examined. Typically, 18 samples cover frequencies from 0–16.9Bark (0–5kHz) in 0.994 Bark-steps[9].

Equal-loudness Preemphasis

In this step, each spectral component is multiplied by a weight given by its frequency f in Hertz[10]:

$$E(f) = \frac{(f^2 + 1.44 \cdot 10^6) f^4}{(f^2 + 1.6 \cdot 10^5)^2 (f^2 + 9.61 \cdot 10^6)}$$

This accounts for the frequency sensitivity of the human ear.

Intensity-loudness Power Law

The last step before computing LPC is to take the cubic root of the pre-emphasized spectrum obtained in the previous step. This as an approximation of the power law of hearing and simulates the relationship between intensity of a sound and the perceived loudness.

Inverse Discrete Fourier Transform

After the aforementioned changes have been applied to the spectrum, the inverse discrete Fourier transform (the size of which is usually 34[9]) is taken to bring the signal back to the time domain. From this signal, the LPC coefficients are computed as described in the previous section. In some applications the PLP coefficients are transformed into cepstral coefficients (PLP-CC)[1].

1.6 Measuring Level of Similarity Between Speech Signals

When building a speech recognition system, we need a way of determining the level of similarity between speech signals. We will represent the signals as series of speech recognition features. The individual features can be seen as points in an N-dimensional vector space, where N is the length of the feature vector.

1.6.1 Similarity of Feature Vectors

To determine how similar two feature vectors are, we compute their distance in the vector space. The lower the distance, the more similar they are. Various distance functions can be used, such as the l_1 metric or Euclidean metric. In our application, the Euclidean metric is used. The Euclidean distance between vectors u and v of N dimensions is given by:

$$d\left(u, v\right) = \sqrt{\sum_{i=1}^{N} \left(u_i - v_i\right)^2}$$

1.6.2 Dynamic Time Warping

Another problem that we encounter is that two speech signals containing the same phrase obtained on two occasions are rarely of the same length. A more precise formulation of the problem follows.

For a feature vector sequence X of M feature vectors and a feature vector sequence Y of N vectors, we want to find the *best alignment*. An alignment of two sequences is defined as a sequence of tuples $p_i = (a, b)$ where $0 \leq a < M$ and $0 \leq b < N$. Let L be the length of the alignment and $p_i = (m_i, n_i)$ the $i + 1$th element of the alignment, then the alignment has to satisfy the following conditions:

- $p_0 = (0, 0)$ and $p_{L-1} = (M - 1, N - 1)$.
- $m_0 \leq m_1 \leq \ldots \leq m_{L-1}$ and $n_0 \leq n_1 \leq \ldots \leq n_{L-1}$
- $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for $0 \leq l < L - 1$

We seek to minimize the cost of the alignment, which is defined as:

$$c_{min} = \sum_{i=0}^{L-1} d\left(X[m_i], Y[n_i]\right)$$

where d is a distance function.

For two sequences there are many possible alignments. It is unfeasible to compute the cost of all of them. Fortunately, there is an algorithm based on dynamic programming, that lets us find the optimal alignment efficiently. An informal description of the algorithm follows, for a detailed description refer to[11].

The Dynamic Time Warping Algorithm

We define $D(m,n)$ as the optimal alignment of subsequences $X(0:m)$ and $Y(0:n)$. These values define an $M \times N$ matrix. Clearly, the optimal alignment for sequences X and Y is $D(M-1, N-1)$. We compute $D(M-1, N-1)$ gradually, using optimal solutions for subsequences of X and Y. First we initialize the matrix with the following values:

$$D(m,0) = \sum_{k=0}^{m} d(x_k, y_0) \text{ for } m \in [0 : M-1]$$
$$D(0,n) = \sum_{k=0}^{n} d(x_0, y_k) \text{ for } n \in [0 : N-1]$$

then we compute the remaining values in the matrix using the following identity:

$$D(m,n) = \min\{D(m-1,n-1), D(m,n-1), D(m-1,n)\} + d(x_m, y_n)$$

for $0 < m < M$ and $0 < n < N$. The optimal alignment of the whole sequences $D(M-1, N-1)$ can be computed in $\mathrm{O}\big(MN\big)$ time[11].

1.7 Voice Activity Detection

Another problem we face when processing a speech recording for speech recognition is the fact that the speech itself rarely spans the whole recording. In other words, there is usually silence at the beginning and at the end of the recording. Our task is to determine when the speech starts and ends in the recording.

Several voice activity detection (VAD) approaches of various robustness have been developed[12][13]. If a speech recognition system is to be used in a noisy environment, the VAD algorithms have to deal with noise detection and suppression. We expect, however, that a voice controlled text editor will be used in a calm working environment. Therefore a simple approach described below is sufficient:

1. Take the first 200 ms long window from the input and compute the average energy (amplitude squared) in that window. This is what we consider to be silence, or "non-speech". This threshold will be used to determine whether a window contains speech or not.

2. Take the next window, starting 100 ms later than the previous and compute the average energy in it. If the average energy is at least twice the value of the threshold, the window is marked as the beginning of the speech in the recording.

3. After finding the start, we determine where the speech ends by repeating the above process, but starting at the end and moving towards the start in time. Once the average energy is twice as high as the threshold, we mark the window as the end of the speech.

Experiments indicate that this method of VAD shows good results, is computationally efficient, but heavily depends on the choice of the threshold[14].

2. Related Works

This chapter provides an overview of currently used speech recognition systems and techniques.

2.1 Types of Speech Recognition Systems

There are three types of speech recognition (SR) systems:

- Speaker-dependent systems - the user is required to dictate every word he or she intends to use with the system. The set of recognizable phrases can be augmented by connected-word recognition which allows for a sequence of phrases from the vocabulary to be recognized (e.g. phone numbers).
- Speaker-independent systems - the system is trained on large amounts of data and does not require any configuration by the user.
- Speaker-adapting systems - a speaker-independent system is adjusted to the user's voice

2.1.1 Speaker-dependent Systems

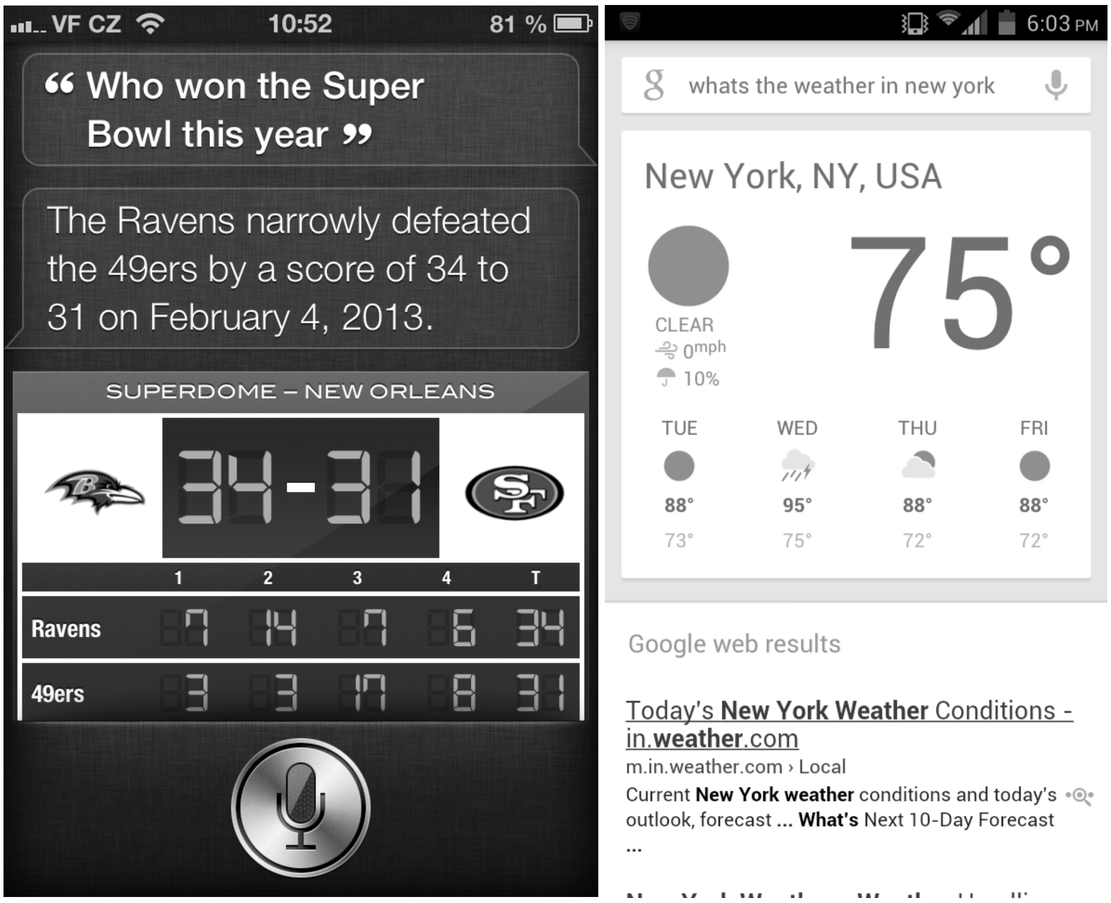
Speaker-dependent systems are scarcely used nowadays. One of the fields where they are still used is warehousing. For example, a solution by *Voxware* allows order pickers to pick orders more quickly, with fewer errors and hands-free. The reason a speaker dependent system is used is, according to the company's web page [17], better performance in a noisy environment.

2.1.2 Speaker-independent Systems

Speaker-independent systems are the most widely used systems. Well-known examples are *Siri*[18] (Figure 2.1a) from Apple and *Google Now*[19] (Figure 2.1b) from Google, which are virtual personal assistant services available on mobile platforms. These services utilize speech recognition, natural language processing and artificial intelligence. A user uses voice input to ask the assistant a question regarding weather, sports results, directions, public transport information and much more. Many third party services are being integrated into the assistants, thus providing table reservations, restaurant reviews and more.

The number of supported input languages is limited and these systems are wont to have problems with accents. This is clear from the fact that users of both services have to choose from several accents of English. Users who don't speak with a "supported accent" often encounter many difficulties when using the software. Although no scientific studies are regarding the problems with accents are available, there are many frustrated users complaining on various web sites, including renowned technology magazines[20].

Furthermore, speaker-independent systems require vast training sets to work well. The processing of the speech commands is carried out on servers, therefore internet connectivity is required. However, this also means that users always use


VF CZ 10:52 81 %

“ Who won the Super Bowl this year ”

The Ravens narrowly defeated the 49ers by a score of 34 to 31 on February 4, 2013.

SUPERDOME – NEW ORLEANS

34-31

	1	2	3	4	T
Ravens	7	14	7	6	34
49ers	3	3	17	8	31

6:03 PM

whats the weather in new york

New York, NY, USA

75°

CLEAR
0mph
10%

TUE	WED	THU	FRI
88°	95°	88°	88°
73°	75°	72°	72°

Google web results

Today's New York Weather Conditions - in.weather.com
m.in.weather.com › Local
Current New York weather conditions and today's outlook, forecast ... What's Next 10-Day Forecast ...

(a) Siri (b) Google Now

Figure 2.1: Virtual personal assistants available on iOS and Android mobile devices

the latest version of the software, which is improving over time with more learning data available.

2.1.3 Speaker-adapting Systems

In case when near-perfect accuracy of speech recognition is required, speaker-adapting systems are used. Upon first using the software, the user is guided through a setup process which fine-tunes the speaker-independent speech recognition system's properties. In this way, a higher accuracy can be achieved[16]. An example of such software is *Dragon NaturallySpeaking*[21]. It provides dictation, text-to-speech and command input.

2.2 Modern Speech Recognition

Modern speech recognition systems consist of three major architectural components:

- Acoustic model
- Language model

- Decoding component

The approach currently employed in vast majority of state-of-the-art SR systems is statistical SR. It can be described by the fundamental equation of statistical SR:

$$\hat{W} = \arg\max_{W} \left(P\left(W\right) P\left(X|W\right) \right)$$

where $X = X_1 X_2 \ldots X_n$ is a feature vector sequence and the goal is to find a word sequence $\hat{W} = w_1 w_2 \ldots w_m$ that has the highest probability. $P\left(W\right)$ and $P\left(X|W\right)$ represent probabilities computed by an acoustic model and a language model, respectively.

2.2.1 Acoustic Models

Creating an acoustic model for SR typically entails building statistical representations of feature vectors obtained from input. One of the most commonly used acoustic models are Hidden Markov Models[22]. Major progress has been achieved recently with neural networks used for acoustic modeling[23].

Pronunciation modelling is also a part of acoustic modelling. It describes how a sequence of basic speech units forms larger speech units, such as words and phrases.

2.2.2 Language Models

A language model is either a grammar describing permissible structures for the language or a stochastic language model. The latter language model type can be defined as a probability distribution over word strings W that reflects how often W occurs as a sentence. For a language model to work well, it necessitates training data consisting of millions of words[24].

2.2.3 Decoding

Decoding is the process of finding a sequence of words whose corresponding acoustic and language models best match the feature vector sequence obtained from input. It is often referred to as a search process. Graph search algorithms serve as a foundation for search algorithms in SR. Speech recognition search is mostly done using the Viterbi decoder[25]. Different language models have great impact on the search complexity.

2.3 Summary

Most modern speech recognition systems are speaker-independent and based on statistical SR. These require large amounts of training data to work well. For many languages, data sets of this size are not available, and in some cases never will be. Moreover, people who speak with a strong accent frequently encounter difficulties when using speaker-independent SR.

The aforementioned drawbacks of speaker-independent systems render speaker-dependent SR systems a viable solution for context-constrained applications in languages used by smaller number of speakers.

3. Implementation

This chapter provides a description of a .NET voice control class library implementation. The implementation uses a feature extraction tool called openSMILE, an overview of such tools is given as well. Description of the programming interface of this library is also provided.

3.1 Available Speech Recognition Feature Extraction Tools

3.1.1 MARSYAS

MARSYAS[26] (Music Analysis, Retrieval and Synthesis for Audio Signals) is a framework for audio analysis. Although its focus is music information retrieval, its capabilities include MFCC extraction. It is available under the GNU Public Licence (GPL) Version 2, for non-commercial use. It is available for Linux, OS X and Windows.

3.1.2 Yaafe

Yaafe[27] (Yet Another Audio Feature Extractor) is a feature extraction toolbox providing several audio features, including MFCC and LPC. It is available under the GNU Public Licence (GPL) Version 3 for Linux and OS X platforms.

3.1.3 openSMILE

OpenSMILE: The Munich Versatile and Fast Open-Source Audio Feature Extractor[29] is a toolbox for extracting both music and speech recognition features. These include MFCC, LPC and PLP features. It is available under the GNU Public Licence and for Linux, OS X and Windows platforms. This toolbox is used in our .NET library, which is described below.

3.2 .NET Class Library Implementation

3.2.1 Using openSMILE to extract speech features

OpenSMILE is available as a command line executable and a C++ library. We use the command line executable, *SMILExtract.exe*, due to the fact that our library is written in C#.

Configuration

The feature extraction process in openSMILE is configured using configuration files. In the configuration file, the user specifies modules to use. These modules perform various tasks, ranging from dividing the signal into frames, applying windows and calculating the Fast Fourier Transform to writing the resulting features to an output file. Each module contains *dataReader* and *dataWriter* components. Using these components, we specify the name of the memory level to which a module should write and from which it should read data. In this way we can connect modules that form a “chain”. We start with a module reading a WAV file containing speech signal, process the signal using the necessary modules and eventually use a module that writes the computed features to a CSV file. Each module has a set of parameters that we use to adjust the extraction process. For instance, the module which divides the signal into frames has parameters that specify how long the frames are and how long an overlap is between two successive frames. More details about modules and their parameters can be found in the openSMILE documentation[28]. Our class library contains a configuration file for each speech recognition feature and its modifications, if used.

Extraction

Once we have created a configuration file, we can use the *SMILExtract.exe* executable to extract features from a WAV file containing speech. To do this, we simply run the executable, specifying 3 parameters:

- Configuration file: -C *configFile*
- Input (WAV) file: -I *inputFile*
- Output (CSV) file: -O *outputFile*

A sample execution may look like:

```
SMILExtract.exe -C myConfig.config -I speech.wav -O features.csv
```

To use a WAV file as an input and a CSV file as an output, it is necessary to specify this in the configuration file. Details can be found in the openSMILE documentation or in the configuration files included with our class library.

Querying

To determine the best matching command in a database, the dynamic time warping algorithm is used. When we query the database with a command, a sequence of features is obtained. This sequence is then aligned with every command present in the database using the DTW algorithm. In the case of speaker-dependent command database, the result of the query is the command from the database which has the lowest alignment cost.

In the case of speaker-independent command database consisting of N speakers, the query algorithm is slightly different. We also compute the DTW alignment cost for each command in the database. Once we have the costs, we select the top N commands with the lowest alignment cost. We look at the names of the commands in this set and return the one that has the highest number of occurrences in the top N matching commands. If there are multiple “top” commands, we choose the one which has the lowest cost.

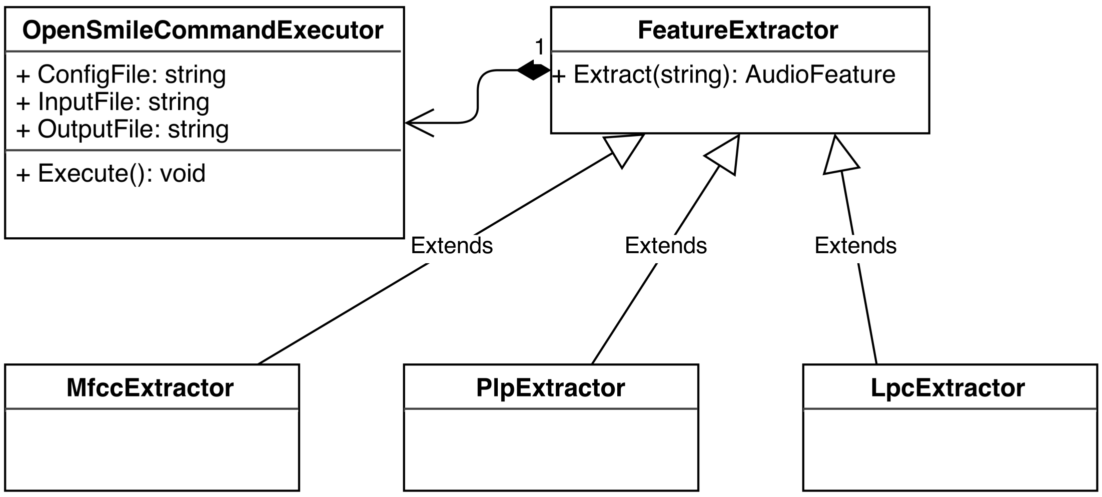


Figure 3.1: Feature extraction class diagram

3.2.2 Object Model

Overview

The class providing interface for voice control is called *VoiceControl*. For recording sound it requires an instance of a class implementing the *IVoiceRecorder* interface. In our case, the *VoiceRecorder* class is used. *VoiceControl* also uses a *VoiceCommandDB* to store commands and find a matching command for speech signals received from the user.

The *VoiceCommandDB* only stores features extracted from a speech signal using one of the classes derived from the *FeatureExtractor* class. The extractor classes utilize the *OpenSmileCommandExecutor* to extract features.

The extracted features are represented by the *AudioFeature* class. The features are stored together with a name assigned to them in the *VoiceCommand* class, which represents a voice command. *VoiceCommand* instances are used inside the *VoiceCommandDB* class.

Feature Extraction Classes

A class diagram of feature extraction classes is shown in figure3.1.

The *OpenSmileCommandExecutor* uses the openSMILE command line executable to extract features. Its three properties, *ConfigFile*, *InputFile* and *OutputFile* are set in the constructor. To extract features from the specified file, using the specified configuration, the *Execute()* method is called.

The *FeatureExtractor* class is an abstract base class for three extractor classes. Each of the three classes is used to extract different speech recognition features. When an instance of an extractor class is created, the corresponding OpenSmileCommandExecutor instance is created and used by the class for feature extraction.

Voice Control Classes

Figure 3.2 contains a class diagram of voice control classes.

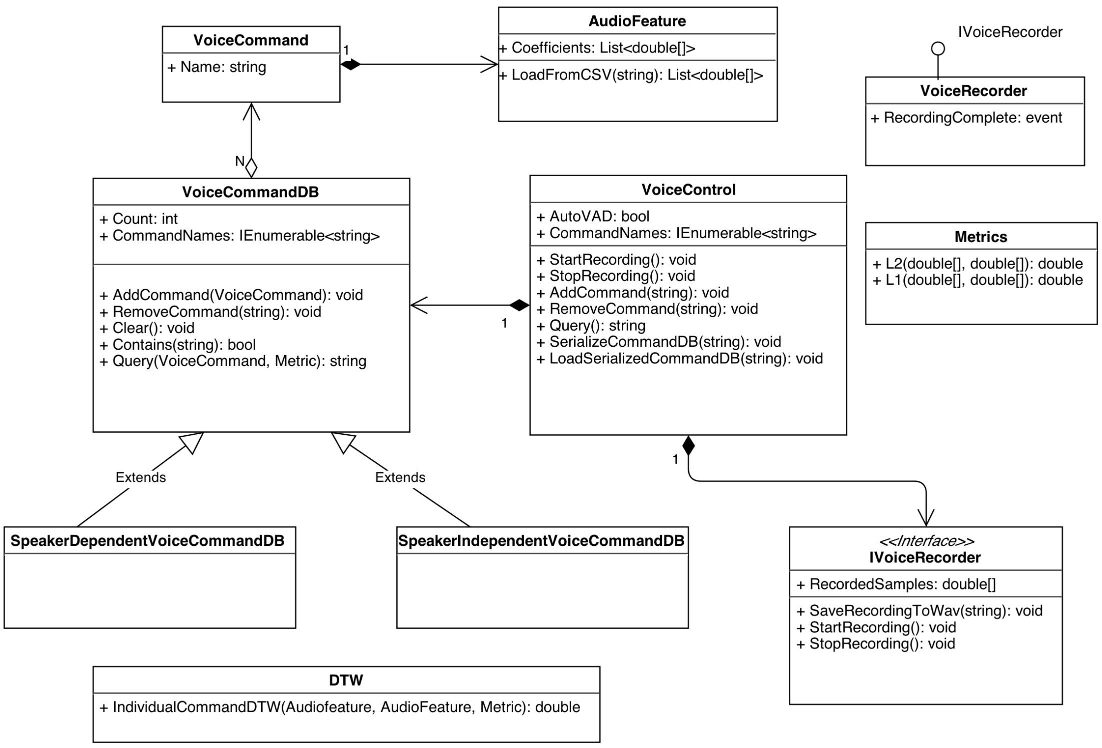

VoiceCommand
+ Name: string
AudioFeature
+ Coefficients: List<double[]>
+ LoadFromCSV(string): List<double[]>
IVoiceRecorder
VoiceRecorder
+ RecordingComplete: event
1
N
VoiceCommandDB
+ Count: int
+ CommandNames: IEnumerable<string>
+ AddCommand(VoiceCommand): void
+ RemoveCommand(string): void
+ Clear(): void
+ Contains(string): bool
+ Query(VoiceCommand, Metric): string
VoiceControl
+ AutoVAD: bool
+ CommandNames: IEnumerable<string>
+ StartRecording(): void
+ StopRecording(): void
+ AddCommand(string): void
+ RemoveCommand(string): void
+ Query(): string
+ SerializeCommandDB(string): void
+ LoadSerializedCommandDB(string): void
Metrics
+ L2(double[], double[]): double
+ L1(double[], double[]): double
1
1
Extends
Extends
SpeakerDependentVoiceCommandDB
SpeakerIndependentVoiceCommandDB
<<Interface>>
IVoiceRecorder
+ RecordedSamples: double[]
+ SaveRecordingToWav(string): void
+ StartRecording(): void
+ StopRecording(): void
DTW
+ IndividualCommandDTW(Audiofeature, AudioFeature, Metric): double

Figure 3.2: VoiceControl object model

The *AudioFeature* class contains the feature coefficients and a method for loading them from a CSV file.

To represent a voice command, the *VoiceCommand* class is used, which contains speech recognition features of a command as well as its name. The name of the command is used for identification.

The process of recording speech is carried out by the *VoiceRecorder* class, which implements the *IVoiceRecorder* interface. This interface declares methods for starting and ending the recording process as well as a method for saving the recorded speech to a WAV file, which is necessary for subsequent feature extraction. Apart from the interface methods, the *VoiceRecorder* also contains an event, *RecordingComplete*. The event is used when automatic voice detection is used. When the recorder obtains a recording, it saves it to a WAV file and uses this event to notify the subscribers about the new recording.

The static *DTW* class contains a method which uses the dynamic time warping algorithm to compute the cost of the optimal alignment between two sequences of audio features. In our case, one sequence is stored in the command database and the other is a sequence obtained from the user's spoken command.

The static *Metrics* class contains methods that compute L1 or L2 distance between two vectors. These are used in the DTW algorithm in the previous class.

The classes extending the *VoiceCommandDB* class provide functionality that allows us to add and remove commands as well as query the database for the best match to a command spoken by the user. The *Query()* method returns the name of the command which has the lowest DTW "distance" from the recorded command.

Finally, the *VoiceControl* class provides an interface for building a speech recognition system. The interface is described in detail in the next section.

3.3 Using the Voice Control Library

In this section, we will explain how to use the Voice Control Library in a .NET assembly.

3.3.1 Prerequisites

To successfully use the library, it is necessary to perform the following steps:

- Place the SMILExtract executable and the library openSmileLib.dll in a folder called *OpenSmile* and place this folder into the folder where our program's executable lies.
- Place the configuration files for feature extraction in a folder called *Configs*, which also has to be located in the same folder as the program's executable.
- Add a reference to NAudio.dll in your project. This library is used to record sound. It is recommended to use the assembly that is supplied with the voice control library.
- Add a reference to VoiceControlLibrary.dll in your project.

3.3.2 Basic usage

After creating an instance of the VoiceControl class using one of the constructors, we can perform the following actions:

- Record a voice command: to start recording, the *StartRecording()* method is used. To end the recording, we use the *StopRecording()* method, which saves the obtained recording to a WAV file called "speech.wav" and extracts audio features specified in the VoiceControl instance. The recorded command's features are temporarily stored until another command is recorded and are used by other methods described in this section.
- Add a voice command to the command database: the *AddCommand(string name)* method is used to add the latest recorded command to the database and assigns the specified name to it. If a speaker-dependent voice command database is used, the name has to be unique.
- Remove a command from the command database: the *RemoveCommand(string name)* method removes a command with the specified name assigned to it from the command database.
- Query the command database: the *Query()* method uses the last recorded command to find the closest match in the command database. It returns a string identifying the best matching command in the database. There is also an overload of the Query() method, which provides an output parameter, through which we obtain the DTW cost of the match.
- Save the command database for later use: we can persist the recorded command database using the *SerializeCommandDB(string fileName)* method. To load a persisted database, the *LoadSerializedCommandDB(string fileName)* method is used.

3.3.3 Advanced Usage

It is also possible to specify what speech recognition features the voice control should use as well as toggle automatic voice detection. There are three constructors apart from the default one that facilitate customization:

- *VoiceControl(RecordingObtainedEventHandler handler)*: this constructor is the same as the default one, except that it sets the AutoVAD property to true and assigns the handler from the argument to its VoiceRecorder's RecordingComplete event. Thus, it is possible to be notified when a new recording becomes available.

- *VoiceControl(IVoiceRecorder recorder, FeatureExtractor extractor, VoiceCommandDB db, Metric metric)*: this constructor allows us to specify the features supplying a FeatureExtractor instance to be used. We can also supply our own database, a voice recorder and choose a metric that should be used in Dynamic time warping.

- *VoiceControl(IVoiceRecorder recorder, FeatureExtractor extractor, VoiceCommandDB db, Metric metric)*: the constructor is the same as the one above, but allows us to assign a RecordingComplete event handler. As with the first constructor, the AutoVAD property is set to true.

To turn on automatic voice detection, we set the *AutoVAD* property to true.

4. A Voice Controlled Text Editor

In this chapter, we examine the task of building a voice controlled text editor. An implementation of a simple voice controlled editor, which utilizes the library described in the previous chapter, is presented.

4.1 Motivation

Since the advent of the personal computer, text editors have gained many features. The way we interact with these programs has not changed significantly, however. Together with the number of features grew the number of icons, sliders and other user interface (UI) elements available to the user. This has made the learning curve steeper and hidden many useful functions from a casual user.

Moreover, if a user has to constantly click on buttons to make even the slightest changes, his or her work flow is hindered. The issue is exacerbated for people who do creative writing. Of course, one can use keyboard shortcuts in many cases to alleviate this problem. This, however, requires the user to learn the shortcuts and it is rarely the case that a user remembers more than few shortcuts.

As opposed to keyboard shortcuts and a large set of multi-layered menus, buttons and other UI elements, controlling the program using speech is natural. If we allow the user to record their own command for each action, they do not have to memorize anything, they just tell the program what to do. What is more, it allows the user to concentrate on his or her work, which leads to greater productivity.

4.2 A List of Possible Commands

We used a collaboratively edited online spreadsheet to determine the set of commands that a voice controlled text editor could incorporate. The set of commands is listed below.

1. Page Up/Down
2. Copy, Paste, Cut
3. Undo, Redo
4. Select line
5. Select paragraph
6. Select all
7. Numbers 1-19
8. Numbers 20, 30, . . . , 100
9. Go to the beginning
10. Go to the end of file

11. Find
12. Next - jumps to the next occurrence of the searched word
13. Previous - jumps to the previous occurrence of the searched word
14. Find and replace
15. Open file
16. Save file
17. Save file as
18. Select from here left - selects text starting at the cursor position and ending at the beginning of the line
19. Select from here right - selects text starting at the cursor position and ending at the end of the line
20. Insert page break
21. Newline
22. Minimize
23. Maximize
24. Snap the window to the left
25. Snap the window to the right
26. Print
27. Save as PDF
28. Send via email
29. Compare
30. Insert a link
31. Insert a header
32. Insert a footer
33. Insert a table
34. Insert a horizontal line
35. Insert an image
36. Insert an equation
37. Spell check
38. Set the document language

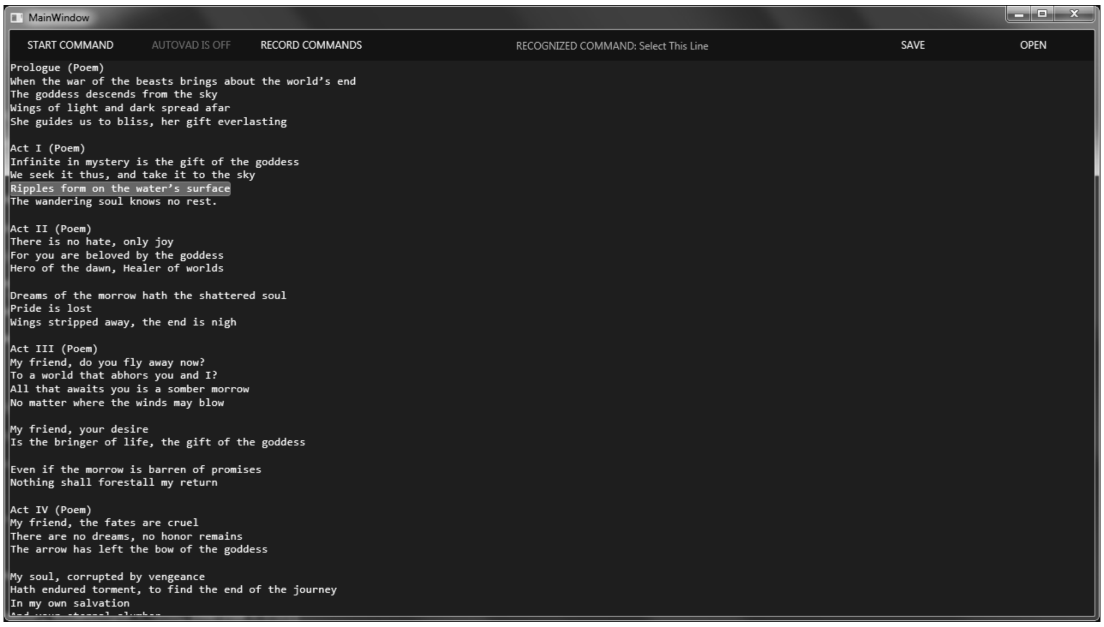
39. Show information - shows document properties, e.g.: word count
40. Help
41. New window
42. New document
43. Set font size
44. Bold
45. Underline
46. Italic
47. Align left
48. Align right
49. Center
50. Justify
51. Show the character map
52. Scroll - starts scrolling until the user stops it
53. Switch to the next document - when multiple files are open, switches to the next document

Most of the commands are either frequently used actions or actions that are usually not easily accessible for a novice user. For example, allowing the user to save his or her work by merely saying “save” can save hours of work for forgetful users. On the other hand, commands such as “show the character map” are used sparsely, but are quite difficult or tedious to access.

These commands are used in a speaker-independent command database to assess the performance of speech recognition features. A subset of the commands is used to query the database and determine the success rate of the evaluated features.

4.3 A Simple Voice Controlled Text Editor

To provide an example of how the class library from the previous chapter can be used to implement a speech recognition system, we present a simple voice controlled text editor. Its capabilities only include a subset of commands from the aforementioned list, since it is beyond the scope of this thesis to implement a more complex solution.



```
Prologue (Poem)
When the war of the beasts brings about the world's end
The goddess descends from the sky
Wings of light and dark spread afar
She guides us to bliss, her gift everlasting

Act I (Poem)
Infinite in mystery is the gift of the goddess
We seek it thus, and take it to the sky
Ripples form on the water's surface
The wandering soul knows no rest.

Act II (Poem)
There is no hate, only joy
For you are beloved by the goddess
Hero of the dawn, Healer of worlds

Dreams of the morrow hath the shattered soul
Pride is lost
Wings stripped away, the end is nigh

Act III (Poem)
My friend, do you fly away now?
To a world that abhors you and I?
All that awaits you is a somber morrow
No matter where the winds may blow

My friend, your desire
Is the bringer of life, the gift of the goddess

Even if the morrow is barren of promises
Nothing shall forestall my return

Act IV (Poem)
My friend, the fates are cruel
There are no dreams, no honor remains
The arrow has left the bow of the goddess

My soul, corrupted by vengeance
Hath endured torment, to find the end of the journey
In my own salvation
```

Figure 4.1: The text editor window

4.3.1 Usage Guide

Figure 4.1 shows the main window of the editor. There are five buttons at the top, from right to left:

- Start Command: When automatic voice activity detection is disabled (see the next button description), this button starts recording a voice command. When the button is clicked again, the recording is stopped, features extracted and the best match from the database is used to execute the corresponding command.

- Automatic Voice Activity Detection Toggle: When this button is clicked, spoken commands are detected automatically. After obtaining a recording, the process is the same as in the previous case.

- Record Commands: this button displays a pop-up window (Figure 4.2) which allows the user to record his or her commands for various actions, as well as delete existing ones in order to re-record them. The commands that are recorded are indicated by green color.

- Save: saves the edited file

- Open: opens a text file

The block of text in the middle of the top bar displays the name of the last matching command found in the database, i.e. the result of the last query.

4.3.2 Implementation

Commands and the corresponding actions are stored in the program using *Dictionary<string,Action>*, where the key is the command name (the same as in the command database) and *Action* is a delegate representing a function with no

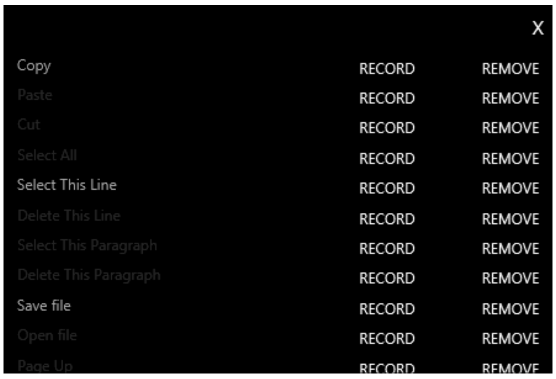


Figure 4.2: Command recording pop-up window

parameters and return type of void. When the user speaks a command, the voice control queries the command database and uses the obtained string to execute the action associated with the string in the aforementioned dictionary.

4.4 Further Challenges

The following problems need to be solved in order to build a robust and reliable solution:

- Noise suppression - the performance of a speech recognition system can be severely affected by noise environment, e.g.: noise from the street. A robust solution must eliminate noise before adding recorded commands to the database and before querying the database with a command from the user. An approach for noise estimation is described for example in [30].
- Voice activity detection - the approach for voice activity detection presented in Section 1.7, is only usable in calm environments. A system using this approach can, for example, easily mistake a dog bark for a voice command. To determine whether the sound recorded is indeed a command from the user, a more sophisticated approach has to be employed, such as[12].
- Speaker identification - in an environment where multiple speakers may be present, it is necessary to distinguish between the speakers and only accept commands from the right user.

5. Experiments

This chapter contains an evaluation of speech recognition features described in the first chapter as well as an assessment of a speaker-dependent and a speaker-independent speech recognition using these features.

5.1 Goals

We will evaluate the performance of five speech recognition features as specified in the next section. Apart from assessing the feature performance in a speaker-dependent speech recognition, we will also use a speaker-independent command database consisting of 20 speakers to test the performance of the five features.

5.2 Evaluated Features

The speech recordings used in the test are sampled at 16 kHz and contain one audio channel. The features are extracted using openSMILE. The common extraction parameters for all the features are:

- Frame length = 25 ms
- Frame step length = 10 ms. The next frame starts 10 ms after the previous one. In other words, adjacent frames have a 15 ms long overlap.
- Window type = Hamming window

The features evaluated are:

1. MFCC: the number of coefficients we use is 13, which is the typical value used for MFCC, as mentioned in 1.5.1. Because the number of mel frequency bands has been shown to have insignificant impact on the perfomance in [31], we use the openSMILE default value, which is 26.
2. MFCC + Delta: Delta coefficients as described in 1.5.1 are added to MFCC coefficients.
3. MFCC + Delta + Acceleration: Acceleration coefficients as described in 1.5.1 are added to the MFCC and Delta coefficients.
4. LPC: The choice of LPC order is discussed in [1]. It is suggested to use 1 coefficient for every kHz contained in the speech, plus 2–4 coefficients. The authors also note that if the order is too high, it may lead to worse separation of the source from the vocal tract shaping. That is why we choose the order of LPC to be 10.
5. PLP: It is shown in [9] that the optimal order for PLP is 5, which is the order we use.

5.3 Methodology

We have collected recordings from 5 female and 5 male speakers. Each speaker recorded 24 commands, the list of which is located below. Each of the 24 commands was recorded on two occasions by the speaker. The recording of the second command did not follow immediately after the first, to ensure sample variability. This gives us two recordings for each command and speaker. For each command, we use one recording to store the command in a speaker-dependent database and the other one to query the database and thus determine performance. The second recording is also used to query a speaker-independent database. We have applied the voice activity detection algorithm, described in chapter 1 to remove silence from all recordings.

5.3.1 List of Commands Used

The following set of commands was used for the evaluation. It is a subset of the commands listed in the previous chapter. The chosen commands include frequently used actions as well as commands that share one or more words, to thoroughly test the accuracy of the features.

1. Page up
2. Page down
3. Cut
4. Copy
5. Paste
6. Undo
7. Redo
8. Select line
9. Select paragraph
10. Select all
11. Go to the beginning
12. Go to the end of file
13. Find
14. Next
15. Previous
16. Find and replace
17. Save file
18. Insert a table

Features	**Average Success Rate**
MFCC	69.58%
MFCC + Delta	70%
MFCC + Delta + Acceleration	70.41%
LPC	55.83%
PLP	76.25%

Table 5.1: Average success rates of 5 speech recognition features, for 10 speakers, in a speaker-dependent command database

19. Insert an image
20. Set font size
21. Bold
22. Underline
23. Italic
24. Show the character map

5.3.2 Speaker-dependent Speech Recognition

For each speaker, as discussed in the beginning of this section, we take half of the recordings and create a speaker-dependent database, one for each type of features tested. This leaves us with five databases. We use the second half of the recordings to query the five databases and determine the success rate.

5.3.3 Speaker-independent Speech Recognition

To build a speaker-independent database, we have obtained recordings of all the 85 commands listed in the previous chapter from 20 speakers, 13 of them male and 7 female. The commands were recorded using an Adobe Flash web utility. As in the previous case, we create one database for each type of features. We use the second half of test commands as described in the beginning of this section to query the databases.

5.4 Results

For each speaker, we compute the success rate for each feature type as the ratio between the number of successfully recognized commands and the number of all tested commands.

5.4.1 Speaker-dependent Recognition

Table 5.1 shows the average success rate for the 10 speakers in a speaker-dependent speech recognition system.

Features	Average Success Rate
MFCC	3.75%
MFCC + Delta	3.75%
MFCC + Delta + Acceleration	5%
LPC	1.67%
PLP	12.08%

Table 5.2: Average success rates of 5 speech recognition features, for 10 speakers, in a speaker-independent command database.

5.4.2 Speaker-independent Recognition

Table 5.2 shows the average success rate for the 10 speakers in a speaker-independent speech recognition system.

5.4.3 Recognition Success Rates of Individual Commands

We have also calculated success rates for individual commands. It is given as the ratio of the number of speakers for whom the command was recognized and the total number of speakers. This allows us to see if there is a significant difference between success rates of commands that share words with other commands and those which do not. Table 5.3 shows the success rates of the commands in a speaker-dependent command database.

5.4.4 Extraction Time

Using a set of 85 recordings, we have computed the average extraction time for the tested features. The results are in table 5.5. Each recording used in the test corresponds to one command from the list in chapter 4.

5.5 Discussion

5.5.1 Speaker-dependent speech recognition

The results indicate that adding delta features to MFCC has little effect on success rate. LPC coefficients achieved the poorest results. However, performing perceptual modelling of the spectrum before conducting linear prediction in case of PLP makes it the most accurate of the features. What is more, PLP achieves the best results using far less data to describe a speech signal, compared to other features. While using 39 coefficients in case of MFCC with delta and acceleration coefficients does not provide nearly any performance improvement compared to using 13 MFCC coefficients, PLP manages to outperform all the other features while using only 5 coefficients to describe each frame of the signal.

5.5.2 Speaker-independent speech recognition

In speaker-independent speech recognition PLP again shows significantly better results, compared to the other evaluated features. The overall performance of

Command	**MFCC**	**MFCC+D**	**MFCC+D+A**	**LPC**	**PLP**
Page up	50%	60%	60%	50%	70%
Page down	80%	80%	80%	20%	70%
Cut	90%	90%	90%	80%	90%
Copy	80%	80%	80%	70%	80%
Paste	60%	50%	50%	60%	70%
Undo	60%	60%	60%	60%	60%
Redo	60%	60%	60%	50%	60%
Select line	70%	70%	70%	50%	90%
Select paragraph	60%	70%	70%	40%	80%
Select all	70%	60%	60%	50%	90%
Go to the beginning	80%	80%	90%	50%	80%
Go to the end of file	60%	60%	60%	40%	70%
Find	60%	60%	60%	40%	50%
Next	50%	50%	50%	60%	60%
Previous	60%	70%	70%	70%	90%
Find and replace	80%	90%	90%	70%	70%
Save file	80%	80%	80%	60%	90%
Insert a table	100%	90%	90%	80%	70%
Insert an image	50%	50%	50%	50%	60%
Set font size	80%	80%	80%	80%	100%
Bold	80%	80%	80%	70%	80%
Underline	60%	60%	60%	40%	90%
Italic	60%	60%	60%	50%	60%
Show the character map	90%	90%	90%	50%	100%

Table 5.3: Success rates of individual commands in a speaker-dependent database. **MFCC+D** means MFCC with delta coefficients, and **MFCC+D+A** means MFCC with delta and acceleration coefficients

Command	**MFCC**	**MFCC+D**	**MFCC+D+A**	**LPC**	**PLP**
Page up	20%	20%	20%	0%	30%
Page down	0%	10%	10%	0%	10%
Cut	10%	10%	10%	0%	20%
Copy	0%	0%	0%	0%	10%
Paste	30%	40%	40%	20%	10%
Undo	0%	0%	0%	0%	0%
Redo	0%	0%	0%	0%	20%
Select line	0%	0%	0%	0%	10%
Select paragraph	0%	0%	0%	0%	0%
Select all	0%	0%	0%	0%	10%
Go to the beginning	0%	0%	0%	0%	0%
Go to the end of file	0%	0%	0%	0%	10%
Find	10%	10%	10%	0%	30%
Next	0%	0%	0%	10%	10%
Previous	10%	0%	0%	0%	0%
Find and replace	0%	0%	0%	0%	0%
Save file	0%	0%	0%	0%	30%
Insert a table	0%	0%	0%	0%	0%
Insert an image	0%	0%	10%	0%	0%
Set font size	10%	0%	0%	10%	40%
Bold	0%	0%	20%	0%	20%
Underline	0%	0%	0%	0%	20%
Italic	0%	0%	0%	0%	0%
Show the character map	0%	0%	0%	0%	10%

Table 5.4: Success rates of individual commands in a speaker-independent database. **MFCC+D** means MFCC with delta coefficients, and **MFCC+D+A** means MFCC with delta and acceleration coefficients

Features	**Average Extraction Time**
MFCC	0.176 s
MFCC + Delta	0.179 s
MFCC + Delta + Acceleration	0.191 s
LPC	0.173 s
PLP	0.169 s

Table 5.5: Average extraction time of the 5 features, obtained from 85 extractions from recordings corresponding to the 85 commands listed in chapter 4

the speaker-independent system, however, is poor. There are multiple reasons for this:

- Inconsistency of the obtained recordings: the recordings that are used in the database have been recorded using several different devices of various quality. Although we have removed recordings that were incomplete or unrecognizable by human, we could not account for volume variations and background noise, which may have affected the recognition rate.
- Variations in accent and pronunciation: The commands have been recorded by non-native speakers, among who the variations in accent and pronunciation are larger than among native English speakers.
- Low number of speakers: a larger number of speakers would help account for the large number of variations mentioned in the previous point.
- DTW cost: the DTW cost calculated for the same command spoken by two different speakers can be quite large, as examined in [32]. This causes a large number of mismatched commands.
- Silence removal: in some cases, the algorithm used to remove silence may have removed parts of the spoken command, due to the part being too silent.

The results unequivocally show that using only speech features and DTW is insufficient for a speaker-independent speech recognition system. More sophisticated methods have to be used, such as acoustic modelling described in the second chapter.

5.5.3 Recognition Success Rates of Individual Commands

As we can see from table 5.3, the fact that two commands share a sequence of words does not affect accuracy, i.e. the results for these commands are not significantly worse than for the other commands. For commands which start or end with an unvoiced sound, it is possible that, for some speakers, a small part of the speech has been removed by the voice activity detection algorithm. Examples of such commands are “Find”, where “d” is unvoiced or “Next”, where “xt” is unvoiced. This lowered the success rate, especially for speakers, whose recordings had very narrow dynamic range (the difference between the loudest and the most silent moment in the recording).

5.5.4 Extraction Time

There is only a minor difference between the time of extraction of the features. Notably, of the evaluated features, PLP not only are the most accurate speech recognition features, but their extraction time is also the shortest among the features.

5.5.5 Summary

Since our text editor's voice recognition system is a speaker-dependent one, the results for this system are of most interest to us. Using 10 speakers is only enough to determine the relative performance of the features. We have found out that PLP features perform the best among the assessed features. From the results obtained from just 10 speakers, it might seem that the accuracy is still too low to be usable in a user friendly program. However, if we look at the best success rates obtained for a single speaker, they are well above 90%. Upon closer examination of the records that scored a high success rate, we have found out that the quality of the recording plays a significant part in the accuracy of a speech recognition system. If the user has an average quality recording device (e.g. a built-in laptop microphone), the speech recognition accuracy is very good and the system is usable. In case a user has a recording device that produces a noisy recording, or a recording that is too silent, the performance of the speech recognition system suffers.

Conclusion

In this thesis, we have provided an introduction to digital signal and speech processing. We have explained how sound is digitized in order to be processed by a computer. We use the Fast Fourier Transform algorithm to find the frequency components of a sound signal. We can use this information to describe the process of human speech production. Speech production and perception has been explained in order to better understand the motivation behind various speech recognition features. The speech recognition features are a means of describing a speech signal, so that it can be matched with another speech signal already stored in the database of a speech recognition system. We have introduced three types of features: MFCC, LPC and PLP. To match sequences of features, we use the dynamic time warping algorithm, which allows us to find for a command spoken by the user, the best matching command in the command database. Before we process a recording obtained from the user, we have to remove silence that does not bear any information. To accomplish this, we have described a simple voice activity detection algorithm.

We have also explored available speech recognition systems and the ways in which they are used. An overview of modern state-of-the-art speech recognition techniques has been given as well.

We have created a .NET class library which allows programmers to easily implement a simple speech recognition system in their programs.

To provide an example of how the class library might be used and to show how voice control can be used in a text editor, we have implemented a simple voice controlled text editor.

Finally, we have carried out experiments to determine the accuracy of the presented features in both a speaker-dependent and a speaker-independent system. We have found out that the approach of using speech features and dynamic time warping is insufficient for a speaker-independent system. However, in the case of a speaker-dependent system, the performance of the features is significantly better and provided that the recordings obtained are of sufficient quality, a reliable, speaker-dependent speech recognition system can be implemented using this approach.

Further augmentations may develop techniques of coping with a noisy speech signal or signal recorded in a noisy environment, which would improve the speech recognition system's robustness. Moreover, work effectiveness can be increased with a speaker-independent dictation.

However, not all possible augmentations have to revolve around the speech recognition effectiveness. As voice controlled programs are still a novelty, it is important to develop new user interface paradigms that allow for a seamless experience and workflow.

Bibliography

[1] Huang, Xuedong. *Spoken language processing: a guide to theory, algorithm, and system development.* New Jersey: Prentice-Hall, 2001. ISBN 01-302-2616-5.

[2] Priemer, Roland. *Introductory signal processing.* Teaneck, NJ: World Scientific, c1991, xvi, 734 p. ISBN 99-715-0920-2.

[3] Plannerer, Bernd. *An introduction to speech recognition.* Munich, Germany, 2005.

[4] Rodd, J. M. *The Neural Mechanisms of Speech Comprehension: fM-RI studies of Semantic Ambiguity.* Cerebral Cortex. 2004-11-24, vol. 15, issue 8, p. 1261-1269. DOI: 10.1093/cercor/bhi009. Available at: http://www.cercor.oupjournals.org/cgi/doi/10.1093/cercor/bhi009

[5] *Digital Audio.* In: Digital Audio [online]. 2013 [cit. 2013-07-08]. Available at: http://documentation.apple.com/en/finalcutpro/usermanual/index.html #chapter=52%26section=7

[6] Chittka, Lars and Axel Brockmann. *Perception Space - The Final Frontier.* PLoS Biology. 2005, vol. 3, issue 4. DOI: 10.1371/journal.pbio.0030137.

[7] *Karbos Guide.* Karbos Guide [online]. 2013 [cit. 2013-07-08]. Available at: http://www.karbosguide.com/books/videosound/chapter05.htm

[8] Fastl, Hugo , Eberhard Zwicker. *Psychoacoustics facts and models.* 3rd ed. Berlin: Springer, 2007. ISBN 35-406-8888-9.

[9] Hermansky, Hynek. *Perceptual linear predictive (PLP) analysis of speech.* The Journal of the Acoustical Society of America. 1990, vol. 87, 1738 pages.

[10] Hönig, Florian and Stemmer, Georg and Hacker, Christian and Brugnara, Fabio. *Revising Perceptual Linear Prediction (PLP).* INTERSPEECH, 2005, p. 2997–3000.

[11] Müller, Meinard. *Information retrieval for music and motion.* New York: Springer, 2007, xv, 313 p. ISBN 35-407-4047-3.

[12] Wu, Bing-Fei and Wang, Kun-Ching. *Voice activity detection based on auto-correlation function using wavelet transform and teager energy operator.* Computational Linguistics and Chinese Language Processing. 2006, vol. 11, issue 1, p. 87-100.

[13] Ouzounov, Atanas. *A Robust Feature for Speech Detection.* Cybernetics and Information Technologies. 2004, vol. 4, issue 2, p. 3-14.

[14] Keerio, Ayaz and Mitra, Bhargav Kumar and Birch, Philip and Young, Rupert and Chatwin, Chris. *On Preprocessing of Speech Signals.* International Journal of Signal Processing. 2009, vol. 5, issue 3, p. 216-222.

[15] Sajjan, Sharada C. and C Vijaya. *Comparison of DTW and HMM for isolated word recognition.* International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012). IEEE, 2012, p. 466-470. DOI: 10.1109/ICPRIME.2012.6208391. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6208391

[16] Huang, Xuedong. *A Study on Speaker-Adaptive Speech Recognition.* HLT, 1991.

[17] *Voice Picking Technology 101.* In: Voice Picking Technology 101 [online]. 2013 [cit. 2013-07-14]. Available at: http://www.voxware.com/in-your-world/voice-picking-blog/post/view/single/post/voice-picking-technology-101-better-recognize/

[18] *Siri.* In: Siri [online]. 2013 [cit. 2013-07-14]. Available at: http://www.apple.com/ios/siri/

[19] *Google Now.* In: Google Now [online]. 2013 [cit. 2013-07-14]. Available at: http://www.google.com/landing/now/

[20] *Siri, Why Can't You Understand Me?.* In: Siri, Why Can't You Understand Me? [online]. 2013 [cit. 2013-07-14]. Available at: http://www.fastcompany.com/1799374/siri-why-cant-you-understand-me

[21] emphDragon NaturallySpeaking Home Edition. In: Dragon NaturallySpeaking Home Edition [online]. 2013 [cit. 2013-07-14]. Available at: http://www.nuance.co.uk/for-individuals/by-product/dragon-for-pc/home-version/index.htm

[22] Jelinek, Frederick. *Continuous speech recognition by statistical methods.* Proceedings of the IEEE. 1976, vol. 64, issue 4, p. 532-556.

[23] Hinton, Geoffrey, Li DENG, Dong YU, George DAHL, Abdel-rahman MOHAMED, Navdeep JAITLY, Andrew SENIOR, Vincent VANHOUCKE, Patrick NGUYEN, Tara SAINATH a Brian KINGSBURY. *Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups.* Signal Processing Magazine, IEEE. 2012, vol. 29, issue 6, p. 82-97.

[24] Huang, Xuedong and Deng, Li. *An overview of modern speech recognition.* Handbook of Natural Language Processing. 2010, p. 339-366. Boca Raton, FL, USA: CRC, Taylor and Francis.

[25] Viterbi, Andrew. emphError bounds for convolutional codes and an asymptotically optimum decoding algorithm. Information Theory, IEEE Transactions on. 1967, vol. 13, issue 2, p. 260-269. IEEE.

[26] Tzanetakis, George and Cook, Perry. *Marsyas: A framework for audio analysis.* Organised sound. 2000, vol. 4, issue 3, p. 169-175. Cambridge University Press.

[27] Benoit, Mathieu and Essid, Slim and Fillon, Thomas and Prado, Jacques and Richard, Gaël. *YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software.* ISMIR. 2010 , p. 441-446.

[28] Eyben, Florian, Martin Woellmer and Bjoern Schuller. *openSmile: the Munich open Speech and Music Interpretation by Large Space Extraction toolkit.* 2010. Munich, Germany.

[29] Eyben, Florian, Martin Wöllmer, Björn Schuller. *openSMILE - The Munich Versatile and Fast Open-Source Audio Feature Extractor.* Proc. ACM Multimedia (MM), ACM, Florence, Italy. pp. 1459-1462, 25.-29.10.2010 ISBN 978-1-60558-933-6

[30] Hirsch, HG and Ehrlicher, C. *Noise estimation techniques for robust speech recognition* Acoustics, Speech, and Signal Processing. 1995, vol. 1, p. 153-156. IEEE.

[31] Zheng, Fang and Zhang, Guoliang and Song, Zhanjiang. *Comparison of different implementations of MFCC.* Journal of Computer Science and Technology. 2001, vol. 16, issue 6, p. 582-589. Springer.

[32] Bala, Anjali and Kumar, Abhijeet and Birla, Nidhika. *Voice command recognition system based on MFCC and DTW.* International Journal of Engineering Science and Technology. 2010, vol. 2, issue 12, p. 7335-7342.