Technical University of Crete
Department of Electronics and Computer Engineering

Thesis title: Automatic speech recognition using soft feature decoding

Undergraduate Student:
Tsiartas Andreas

Committee:

Potamianos Alexandros(Supervisor)
Digalakis Vasileios
Liavas Athanasios

Table of Contents

Table of Figures

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Alexandros Potamianos, for his guidance and his support.

I am also very grateful to the members of the supervisory committee.

I would like also to thank my friends and especially my family for their support.

Special thanks to Margarita.

Preface

Nowadays, speech recognition applications are increasing dramatically and predictions show that this tends will continue for the next decade. Already, palmtops have integrated speech recognition applications which are particularly useful in cases in which we cannot use a keyboard. Nowadays, the processing power has been increased significantly and we have the opportunity to integrate speech recognition applications even in cell phones.

Last decade, automatic speech recognition improved greatly and various ASR techniques used today was a product of that decade. Typically, speech recognition starts with the digitized speech signal and this signal is processed in the front-end. The most popular methods to process the signal are the Linear Prediction Analysis and the Mel-Frequency Cepstral Coefficients. A new processing method is the analysis by using formants as features and by using this method we will try to improve the front-end.

Formants are the resonant frequencies of the vocal tract when vowels are pronounced. Also, formants contain useful information in predicting the vowel pronounced but in the presence of noise formant-based recognition is greatly reduced. Our purpose is to benefit from formants properties and improve the recognition performance of the front-end.

Hidden Markov Models provide a popular and effective method to classify the speech signal. In experiments where we recognize phonemes, we used HTK toolkit in order to do HMM-based recognition. Another classifier that we used is the popular K-NN classifier, which was used to classify frames of the testing signal.

Generally, in this thesis, we investigated methods to classify frames or phonemes by using formants and confidence measures. Also, we compared our results with MFCC-based recognition results. Furthermore, we tried methods to normalize the vocal tract length of independent speakers.

1 Introduction

In this chapter, we will see some fundamental aspects of the speech recognition process. Also, we will describe some basic time and frequency characteristics of the speech signal. Another useful aspect that is important and relative to our work, is the classification of speech sounds.

1.1 The speech signal and the process of Speech production, transmission and perception.

The production of speech begins when the talker formulates a message in his mind that he wants to transmit to the listener and he creates the printed text that expressing the words of the message. Then, he converts the message into a language code. Once the language code is chosen the talker must execute a series of neuromuscular commands to cause the vocal cords to vibrate and to shape the vocal tract to create the proper sequence of sounds that will be spoken by the talker.

Then speech signal is propagated to the listener through air and the speech perception process begins. In this process, the listener processes the acoustic signal along the basilar membrane in the inner ear providing a spectrum analysis of the signal. A transduction process converts the spectral signal into activity signals on the auditory nerve which corresponds to a feature extraction process. This activity is converted into language code and finally the message is created.

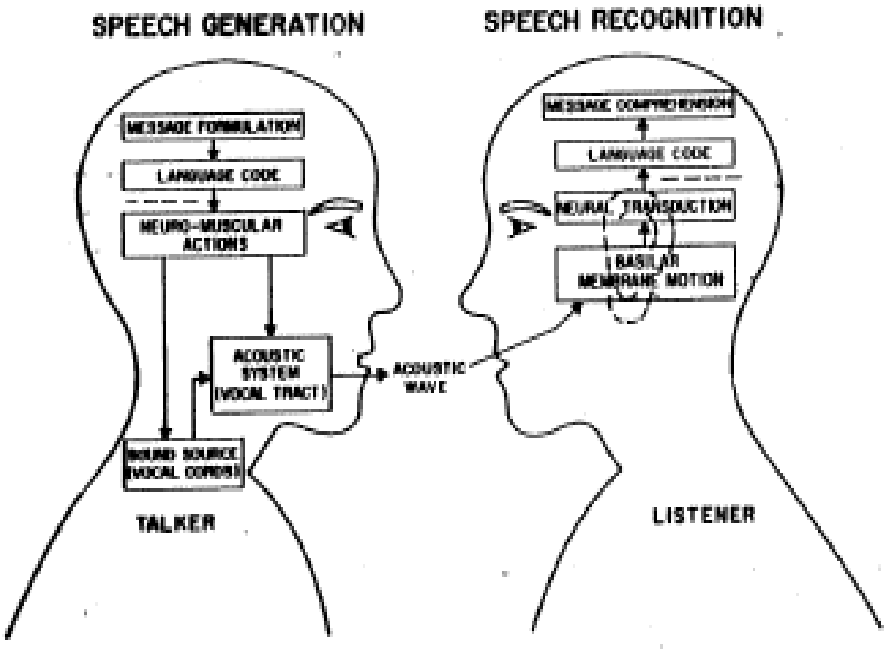

Figure 1. 1 The process of Speech production, transmission and perception**([2])**

1.2 The speech production process.

A schematic diagram of the human vocal mechanism is shown in figure 1.2. Firstly, air enters the lungs through the normal breathing mechanism. As air is expelled from the lungs, vocal cords are caused to vibrate by the air flow. The air-flow is chopped into quasi-periodic pulses which are then modulated in frequency in passing through the throat cavity, the mouth cavity and possibly the nasal cavity. Sounds that are produced depend on the positions of the various articulators (i.e. tongue, lips, mouth, etc).

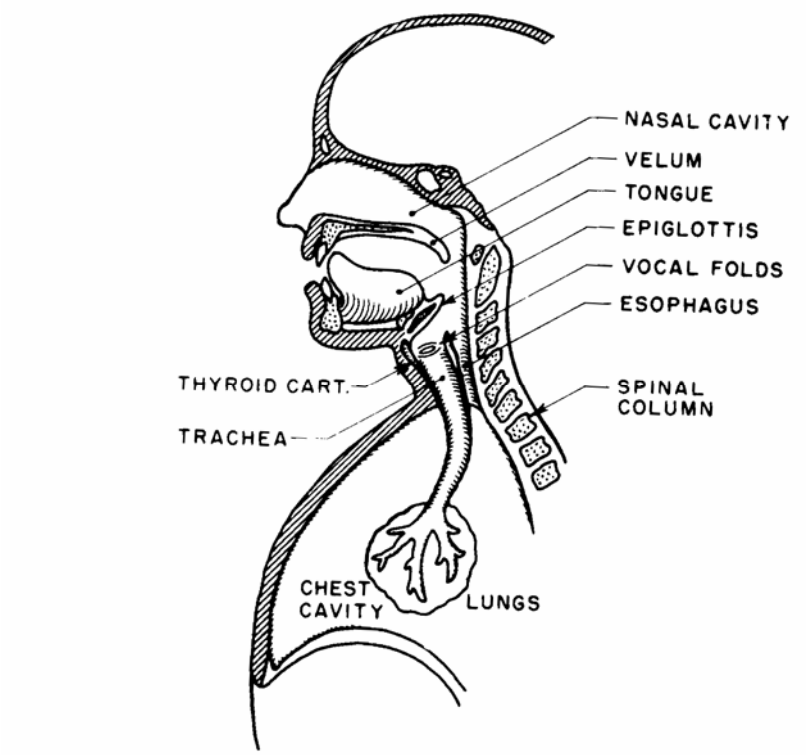

Figure 1. 2 Schematic view of the human vocal tract mechanism([2])

A simplified representation of the complete physiological mechanism of speech creation is shown in figure 1.3. The lungs act as the source of air for exciting the vocal mechanism. The muscle force pushes air out of the lungs and through the bronchi and trachea. Air flow causes vocal cords to vibrate, producing voiced speech signal sounds. When the vocal cords are relaxed, in order to produce sound, the air flow must pass through a constriction in the vocal tract, producing unvoiced sounds.

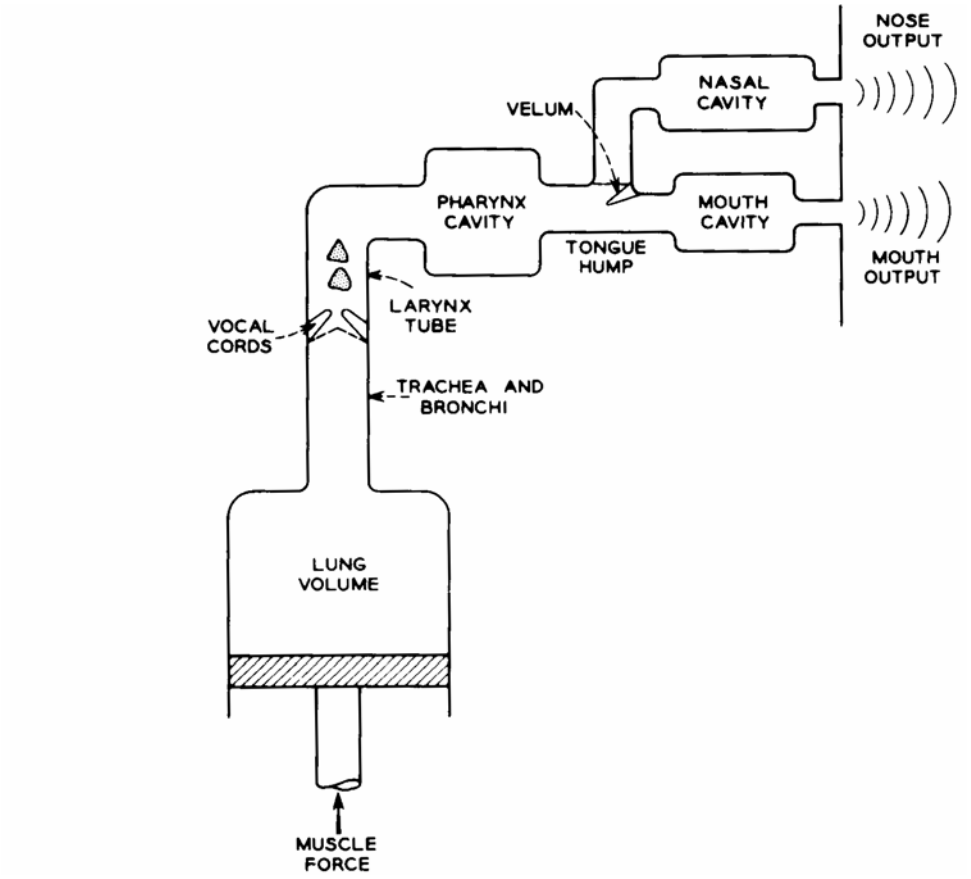

Figure 1. 3 Speech production mechanisms. **([2])**

Speech is produced by a sequence of sounds so the state of the vocal cords as well as the shapes of the various articulators changes over time and speech is produced.

1.3 Speech representation in the time and frequency domains

The speech signal is a time varying signal but when examined over a sufficiently short period, its characteristics are fairly stationary. However, over long periods of time the spectral characteristics change to reflect the different speech sounds being spoken. Figure 1.4 shows the time waveform corresponding to the initial sounds in the phrase "It's time..." Each line corresponds to 100 msec of signal and shows clearly the short period periodicity of the signal.

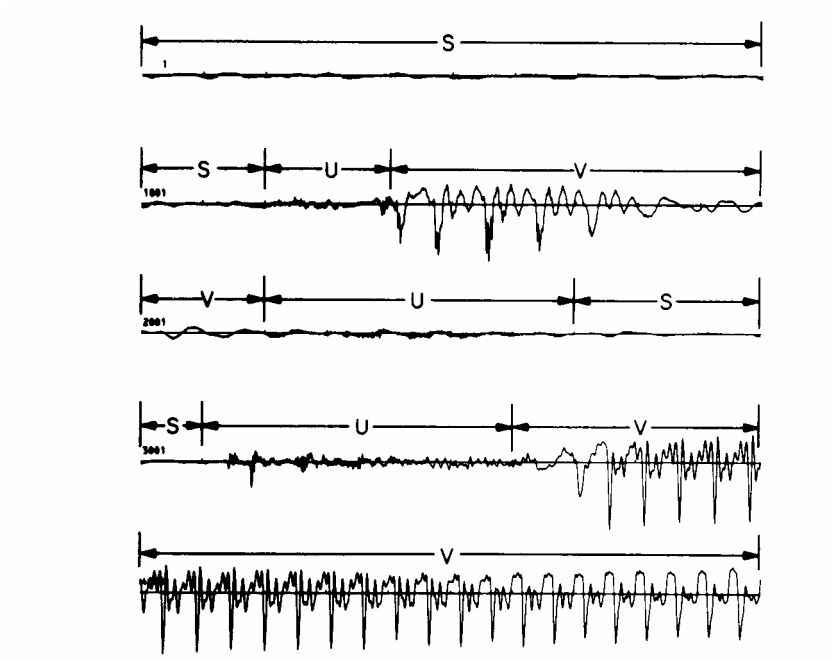

Figure 1. 4 The periodicity of the speech signal([2])

1.4 Classification of speech sounds

A simple way to classify speech is via the state of vocal cords. By using this classification method, we can use a three state representation in which states are silence(where no speech is produced), unvoiced (in this state vocal cords are not vibrating) and voiced ,in which vocal cords are tensed and vibrate periodically when air flows from the lungs, so the speech waveform is quasi-periodic. Figure 1.4 shows this type of classification applied on the above signal.

Another way of classifying the speech signal and representing sound information is via a spectral representation. A well-known representation of this type is the sound spectrogram in which a three-dimensional representation of the speech intensity in different frequency bands, over time is portrayed. Figure 1.5 shows a wideband spectrogram, a narrowband spectrogram and a waveform amplitude panel. The wideband spectrogram corresponds to performing a spectral analysis on 15-msec sections using a broad analysis filter (125 Hz) with the analysis advancing every 1-msec. The narrowband spectrogram corresponds to performing a spectral analysis on 50-msec sections using a broad analysis filter (40 Hz) with the analysis advancing every 1-msec. It is clear that wideband spectrogram provides very good time resolution and narrowband spectrogram provides very good frequency resolution.

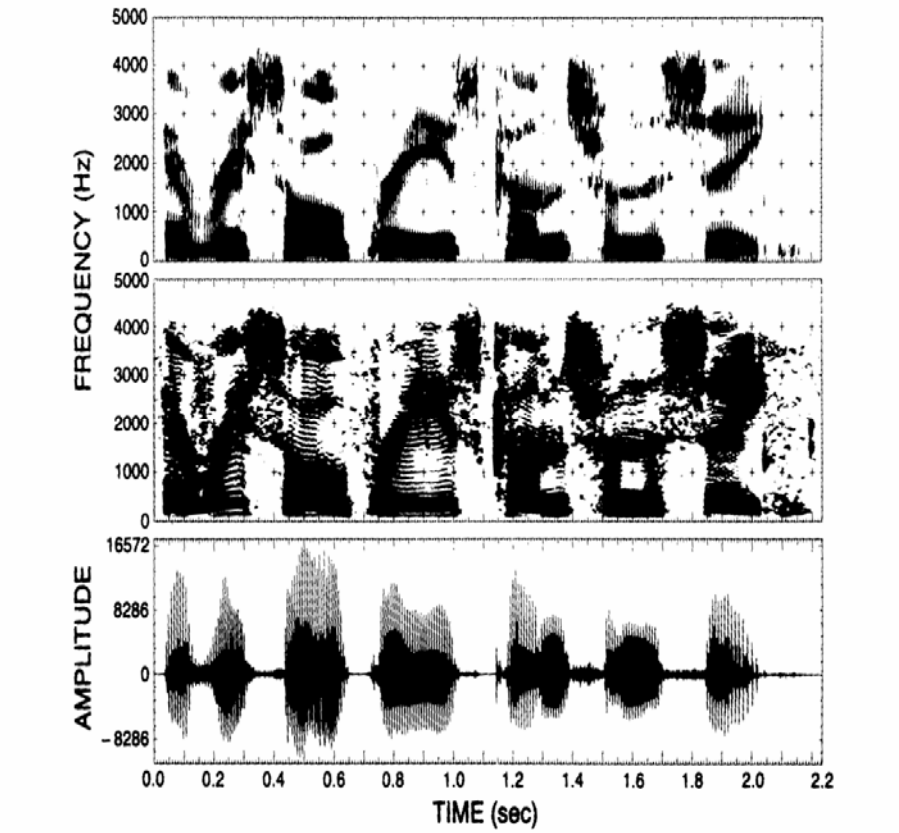

Figure 1. 5 The wideband and narrowband spectrograms and speech amplitude for a speech signal**([2])**

An alternative way of classifying the time-varying characteristics of speech is by using a parameterization of the spectral activity based on the model of speech production described above. Because the human vocal tract is essentially a concatenation of tubes that is excited at the one end or at a point along the tube, we know that the transfer function of energy from excitation source to the output can be described in terms of the natural frequencies or resonances of the tube, called formants. Formants represent the frequencies that pass the most acoustic energy from the source to the output. Figure 1.6 shows a wideband spectrogram along with the formant estimates. This figure clearly shows that there is a good correspondence between the high spectral energy and the formant frequencies.

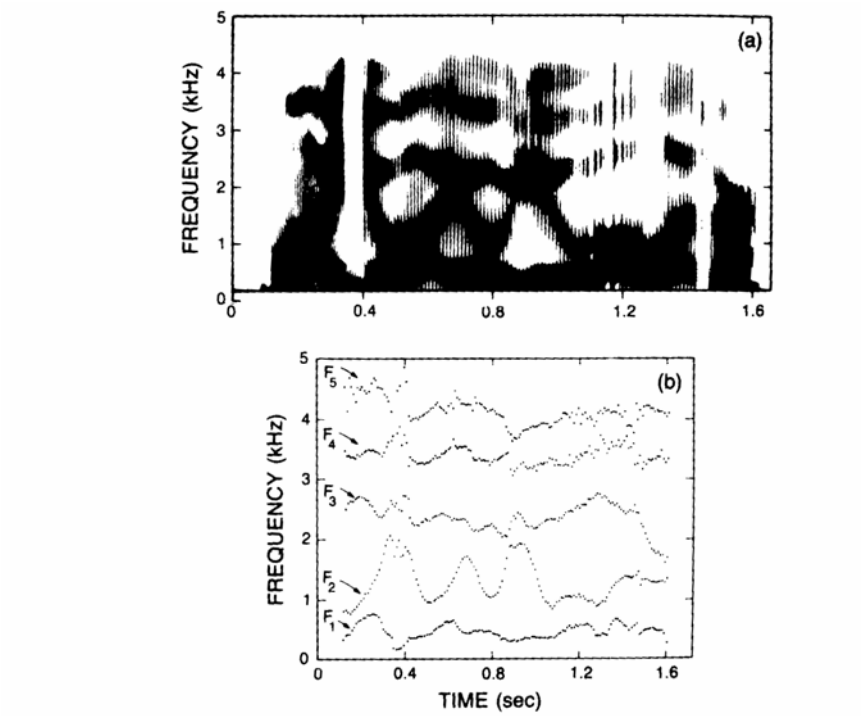

Figure 1. 6 Wideband spectrogram and formant frequency representation of a speech signal**([2])**

1.5 Acoustic phonetics

Most languages can be described in terms of a set of phonemes. For English, there are about 48 phonemes including vowels, diphthongs, semivowels and consonants. A way of studying phonetics that meets our purposes is to consider an acoustic characterization of the various sounds including the place and the manner of articulation, waveforms and spectrographic characterizations of these sounds. Figure 1.7 shows some phonemes and some words that those are used.

Phoneme	ARPABET	Example
/i/	IY	beat
/ɪ/	IH	bit
/e/ (e^{y})	EY	bait
/ɛ/	EH	bet
/æ/	AE	bat
/a/	AA	Bob
/ʌ/	AH	but
/ɔ/	AO	bought
/o/ (o^{w})	OW	boat
/U/	UH	book
/u/	UW	boot
/ə/	AX	about
/ɨ/	IX	roses

Figure 1. 7 Phonetic symbols**([2])**

Phonemes can be classified as either a continuant or a non-continuant sound. Continuant sounds are produced by a fixed vocal tract configuration excited by the appropriate source. The vowels, nasals, voiced and unvoiced fricatives are continuant sounds. All the remaining sounds (diphthongs, semivowels, stops and affricates) are produced by a changing vocal tract configuration, thus they are classed as non-continuants. Figure 1.8 shows the classification of standard English phonemes.

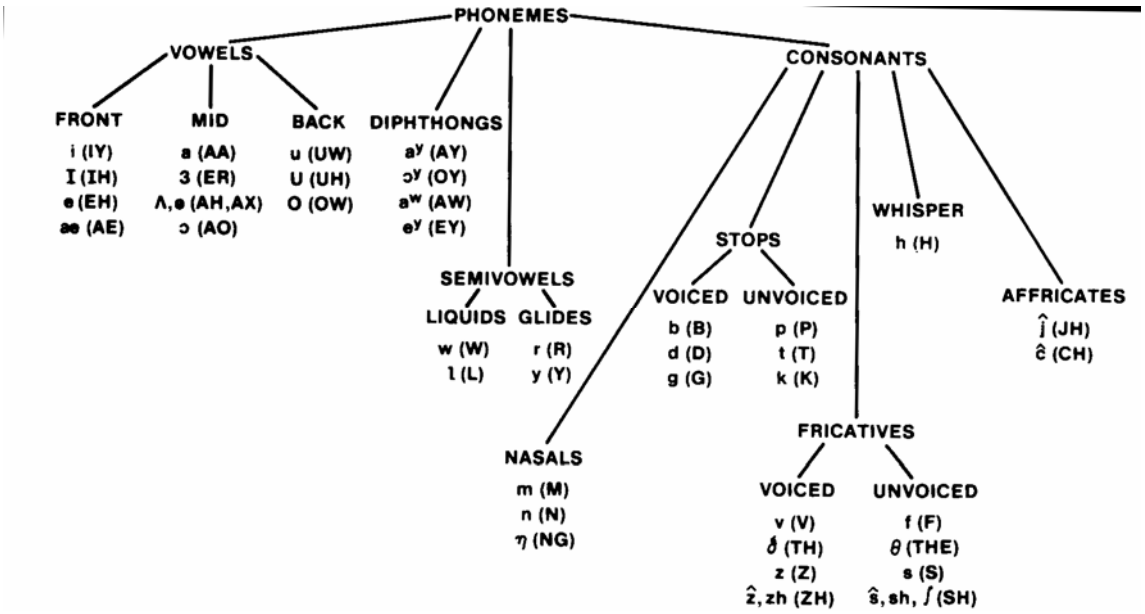

Figure 1. 8 The classification of standard English phonemes**([2])**

1.6 Phonemes production and classification.

1.6.1 Vowels

Vowels are produced by exciting a fixed vocal tract with quasi-periodic pulses of air caused by vibration of the vocal cords. Generally, vowels are long in duration and are spectrally well defined and thus they are usually easily and reliably recognized. As such their contribution to speech recognition system is significant.

There are many ways of classifying vowels, including the typical articulatory configuration required to produce the sounds, including typical waveform plots and spectrogram plots. A simplified way to classify vowel articulatory configuration is in terms of the tongue hump position, for example front, mid, back- and in terms of the tongue hump height, for instance high, mid, low. By saying tongue hump, we mean the mass of the tongue at its narrowest constriction within the vocal tract.

Generally, the front vowels show a pronounced, high-frequency resonance, the mid vowels show a balance of energy over a broad frequency range and the back vowels show a predominance of low-frequency spectral information. A way of exploiting the information embodied in vowels is to represent each vowel by a centroid in the formant space and by this representation we are lead to the vowel triangle which

represents the extremes of formant location in the F1-F2 plane. The vowel triangle with the position of common vowels is shown in figure 1.9

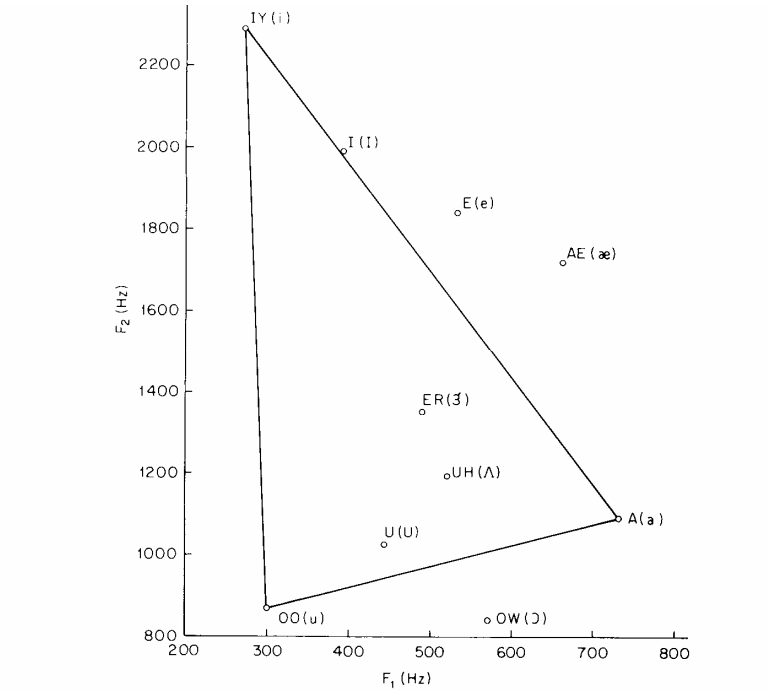

Figure 1. 9 The vowel triangle**([3])**

1.6.2 Semivowels

These sounds are called semivowels because of their vowel-like nature and are characterized by a gliding transition in vocal tract shape between adjacent phonemes. Thus the acoustic characteristics of these sounds are influenced by the context, in which they occur.

1.6.3 Diphthongs

A diphthong is a gliding monosyllabic sound that starts at or near the articulatory position for one vowel and moves to or toward the position for another for another vowel. They are produced by varying the vocal tract smoothly between vowel configurations appropriate to the diphthong and they are characterized by a time varying vocal tract shape (area function) which varies between two vowel configurations.

1.6.4 Nasals

The nasals consonants are produced with glottal excitation and the vocal tract totally constricted at some point along the oral passage-way. The velum is lowered so that air flows through the nasal tract, with sound being radiated at the nostrils. The oral cavity is still acoustically coupled to the pharynx. Thus, the mouth serves as a resonant

cavity that traps acoustic energy at certain natural frequencies. Nasal consonants and nasalized vowels (vowels proceeding or following consonants) are characterized by broader resonances. Nasal consonants are distinguished by the place along the oral tract at which a total constriction is made.

1.6.5 Voiced and unvoiced fricatives

The unvoiced fricatives are produced by exciting the vocal tract by a steady air flow which becomes turbulent in the region of a constriction in the vocal tract. Unvoiced fricative sounds are distinguished by the location of the constriction. The system for producing unvoiced fricatives consists of a source of noise at a constriction which separates the vocal tract into two cavities. Also, fricative excitation is non-periodic.

The voiced fricatives differ from unvoiced fricatives in that two excitation sources are involved in their production. For voiced fricatives the vocal cords are vibrating, and thus one excitation source is at the glottis. Since the vocal tract is constricted at some point forward of the glottis, the air flow becomes turbulent in the neighborhood of the constriction. Thus the spectra of voiced fricatives can be expected to display two components.

1.6.6 Voiced and unvoiced stops

The voiced stop consonants are transient, non-continuant sounds produced by building pressure behind a total constriction somewhere in the oral tract and then releasing the pressure. During the period when there is a total constriction in the tract, no sound is radiated from the lips. Since the stop sounds are dynamical in nature, their properties are highly influenced by the vowel that follows the stop consonant.

The unvoiced stop consonants are similar to voiced stop consonants with one major exception. During the period of total closure of the vocal tract as the pressure builds up, the vocal cords do not vibrate. Then, following period of closure, as the air pressure is release there is a brief interval of friction followed by a period of aspiration before voiced excitation begins.

1.7 Summary

In this chapter, we dealt with some fundamental speech recognition aspects. We saw the signal production and perception process in humans. Moreover, we saw how the signal is represented in time and frequency domain and lastly we described some useful ways of classifying the speech signal.

2 Speech signal processing

In this chapter, we will present some signal analysis techniques that are used in speech recognition systems. Firstly, we will see some signal preprocessing techniques such as windowing and preemphasis filtering. Then, some signal processing techniques will be presented. A brief description of MFCC, LPC and formants analysis will be presented. Moreover, we will describe two vocal tract normalization approaches. The first is based on pitch and the second is the Maximum Likelihood approach to VTLN.

2.1 Short-time Fourier analysis

In order to study spectral properties of speech signals, we need a spectral representation that reflects the time varying properties of the speech waveform. A definition of the time dependent Fourier transform is

$$X_n(e^{j\omega}) = \sum_{m=-\infty}^{\infty} w(n-m)x(m)e^{-j\omega m} \qquad \text{(Eq. 2.1)}$$

In Eq. 2.1, $w(n-m)$ is a real window sequence which determines the portion of the input signal that receives emphasis at a particular time index, n. The time dependent Fourier transform is a function of two variables: the time index, n, which is discrete, and the frequency variable, ω, which is continuous. Thus, the Short-time Fourier transform can be interpreted as a smoothed version of the Fourier transform of the part of the signal within the window.

If the window $w(m)$ has non-zero values for $m = 0,...,L-1$, the frame $x_n(m)$ has non-zero values at

$$x_n(m)\begin{cases}\neq 0 & \text{n-L+1}\le m\le n\\ =0 & \text{else}\end{cases} \qquad \text{(Eq. 2.2)}$$

Thus, the STFT definition for this window type is:

$$X_n(e^{j\omega}) = DTFT\left[x_n(m)\right] = \sum_{m=n-L+1}^{n} x_n(m)\cdot e^{-j\omega m} \qquad \text{(Eq. 2.3)}$$

2.2 Preemphasis filtering

The characteristics of the vocal tract define the current phoneme. Such characteristics are evidenced in the frequency domain by the location of the formants. Although possessing relevant information, high frequency formants have smaller amplitude with respect to low frequency formants. A preemphasis of high frequencies is therefore required to obtain similar amplitude for all formants. Such processing is

usually obtained by filtering the speech signal with a filter whose transfer function in the z-domain is

$$H(z) = 1 - a \bullet z^{-1} \qquad 0.9 \le a \le 1 \qquad \text{(Eq. 2.4)}$$

a being the preemphasis parameter. A typical value for a is 0.95, which gives rise to more than 20 dB amplification of the high frequency spectrum.

2.3 Windowing

Methods for spectral evaluation are reliable in the case of a stationary signal(for example a signal whose statistical characteristics are invariant with respect to time). For voice, this holds only within the short time intervals of articulatory stability, during which a short time analysis can be performed by windowing a signal $x(n)$ into a succession of windowed sequenced $x_t'(n)$, where $t = 1,2,..,T$ called frame, which are then individually processed

$$x_t'(n) = w(n) \bullet x_t(n) \qquad \text{(Eq. 2.5)}$$

The most-used window shape is the hamming window, which has the form

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \qquad 0 \le n \le N-1 \qquad \text{(Eq. 2.6)}$$

2.4 The LPC Model

The basic idea behind the LPC model is that a given speech sample at time n, s(n) can be approximated as a linear combination of the past p speech sample, such that

$$s(n) \approx a_1 s(n-1) + a_2 s(n-2) + ... + a_p s(n-p) \qquad \text{(Eq.2.7)}$$

where the coefficients a1, a2,.. ,ap are assumed constant over the speech analysis frame.

We convert Eq. 2.7 to an equality by including excitation term, G u(n), giving:

$$s(n) = \sum_{i=1}^{p} a_i s(n-i) + Gu(n) \qquad \text{(Eq.2.8)}$$

where u(n) is a normalized excitation and G is the gain of the excitation. By expressing Eq. 2.8 in the z-domain we get the relation

$$S(z) = \sum_{i=1}^{p} a_i z^{-i} S(z) + GU(z) \qquad \text{(Eq. 2.9)}$$

which leads to the transfer function

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 - \sum_{i=1}^{p} a_i z^{-i}} = \frac{1}{A(z)} \qquad \text{(Eq. 2.10)}$$

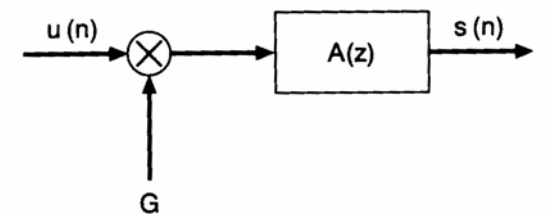

Figure 2. 1 The LP model of speech**([2])**

Figure 2.1 shows the interpretation of Eq. 2.10. The figure shows the normalized excitation source, u(n), being scaled by the gain, G, and acting as input to the all-pole system, $H(z) = \frac{1}{A(z)}$, to produce the speech signal, s(n). Figure 2.2 shows the appropriate synthesis model for speech, corresponding to LPC analysis, in which the excitation function for speech is either a quasi-periodic pulse train or a random noise source for voiced and unvoiced speech sounds respectively. As shown in figure 2.2, the normalized excitation source is chosen by a switch whose position is controlled by the voiced or unvoiced character of the speech, which chooses either a quasi-periodic train of pulses as the excitation for voiced sounds, or a random noise sequence for unvoiced sounds. The gain of the source is estimated from the speech signal, and the source is used as input to a digital filter (H(z)), which is controlled by the vocal tract shape. Thus the parameters of this model are voiced/unvoiced classification, pitch period for voiced sounds, the gain parameter, and the coefficients of the digital filter, {ak}. These parameters vary slowly with time.

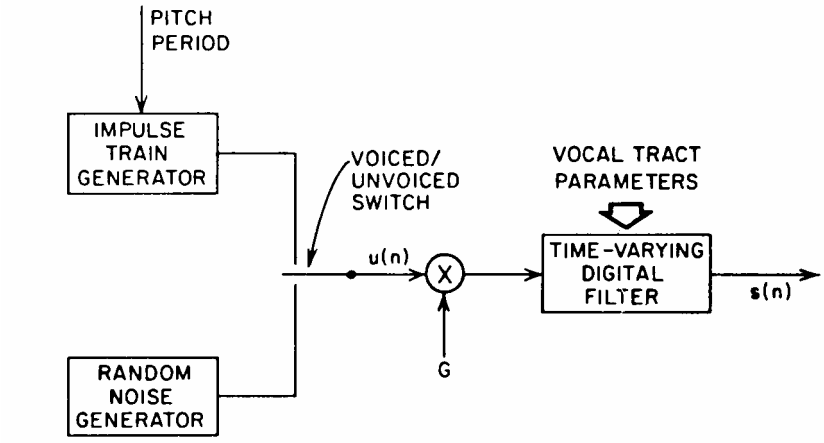

Figure 2. 2 Speech synthesis based on LPC model**([2])**

For the system shown in figure 2.2, the speech samples s(n) are related to the excitation u(n) by the difference equation

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n) \qquad \text{(Eq. 2.11)}$$

A linear predictor with prediction coefficients, a_k is defined as a system whose output is

$$\tilde{s}(n) = \sum_{k=1}^{p} a_k s(n-k) \qquad \text{(Eq. 2.12)}$$

We now form the prediction error, $e(n)$ defined as

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k) \qquad \text{(Eq. 2.13)}$$

From Eq. 2.13 it can be seen that the prediction error sequence is the output of a system whose transfer function is

$$A(z) = \frac{E(z)}{S(z)} = 1 - \sum_{k=1}^{p} a_k z^{-k} \qquad \text{(Eq. 2.14)}$$

Clearly, when s(n) is actually generated by a linear system of the type shown is figure 2.1 then the prediction error, $e(n)$, will equal $Gu(n)$, the scaled excitation.

The basic problem of linear prediction analysis is to determine a set of predictor coefficients {ak} directly from the speech signal in such a manner as to obtain a good estimate of the spectral properties of the speech signal so that the spectral properties of the digital filter of Figure 2.2 match those of the speech waveform within the analysis window. Because of the time varying nature of the speech signal the predictor coefficients must be estimated from short segments of the speech waveform. Thus the basic approach is to find a set of predictor coefficients that minimize this type the mean-squared prediction error over a short segment of the speech waveform.

To find the equations that must be solved to determine the predictor coefficients we define short term speech and error segments at time n as

$$s_n(m) = s(n+m) \qquad \text{(Eq. 2.15)}$$

$$e_n(m) = e(n+m) \qquad \text{(Eq. 2.16)}$$

and we seek to minimize the mean squared error signal at time n

$$E_n = \sum_m e_n^2(m) \qquad \text{(Eq. 2.17)}$$

Which, using the definition of $e_n(m)$ in terms of $s_n(m)$, can be written as

$$E_n = \sum_m \left[s_n(m) - \sum_{k=1}^{p} a_k s_n(m-k) \right]^2 \qquad \text{(Eq. 2.18)}$$

To solve the above equation, for the predictor coefficients, we differentiate E_n with respect to each a_k and set the result to zero,

$$\frac{\partial E_n}{\partial a_k} = 0, \qquad k = 1, 2, ..., p \qquad \text{(Eq. 2.19)}$$

giving

$$\sum_m s_n(m-i) s_n(m) = \sum_{k=1}^{p} a_k \sum_m s_n(m-i) s_n(m-k) \qquad \text{(Eq. 2.20)}$$

By recognizing that terms of the form $\sum s_n(m-i) s_n(m-k)$ are terms of the short-term covariance of $s_n(m)$, i.e.,

we can express Eq. 2.21 in the compact notation

$$\boxed{\phi_n(i,0) = \sum_{k=1}^{p} a_k \phi(i,k)} \qquad \text{(Eq. 2.21)}$$

which describes a set of p equations in p unknowns. The minimum mean squared error, $\hat{E}_n$ can be expressed as

$$\hat{E}_n = \sum_m s_n^2(m) - \sum_{k=1}^{p} a_k \sum_m s_n(m) s_n(m-k) = \phi_n(0,0) - \sum_{k=1}^{p} a_k \phi_n(0,k) \qquad \text{(Eq. 2.22)}$$

Thus the minimum mean-squared error consists of a fixed term $(\phi_n(0,0))$ and terms that depend on the predictor coefficients. To solve (Eq. 2.21) for the optimum predictor coefficients we have to compute $\phi_n(i,k)$ for $1 \le i \le p$ and $0 \le k \le p$, and then solve the resulting set of p simultaneous equations.

2.4.1 The autocorrelation method

The autocorrelation method is a method for determining the limits on the sum in Eq 2.18 and in Eq. 2.20. Firstly, we assume that the waveform segment, $s_n(m)$, is identically zero outside the interval $0 \le m \le N-1$. This can be expressed as

$$s_n(m) = s(m+n)w(m) \qquad \text{(Eq. 2.23)}$$

where w(m) is a Hamming window that is zero outside the interval $0 \le m \le N-1$

If $s_n(m)$ is nonzero only for $0 \le m \le N-1$, then the corresponding prediction error, $e_n(m)$, for a p^{th} order predictor will be nonzero over the interval $0 \le m \le N-1+p$. Thus, E_n can be expressed as

$$E_n = \sum_{m=0}^{N+p-1} e_n^2(m) \qquad \text{(Eq. 2.24)}$$

The limits on the expression for $\phi_n(i,k)$ are the same as the above equation so,

$$\phi_n(i,k) = \sum_{m=0}^{N-1+p} s_n(m-i)s_n(m-k), \qquad \begin{matrix} 1 \le i \le p \\ 0 \le k \le p \end{matrix} \qquad \text{(Eq. 2.25)}$$

or equivalently,

$$\phi_n(i,k) = \sum_{m=0}^{N-1-(i-k)} s_n(m-i)s_n(m+i-k), \qquad \begin{matrix} 1 \le i \le p \\ 0 \le k \le p \end{matrix} \qquad \text{(Eq. 2.26)}$$

Since Eq. 2.26 is only a function of i-k, the covariance function, $\phi_n(i,k)$, reduces to the simple autocorrelation function and becomes,

$$\phi_n(i,k) = r_n(i-k) = \sum_{m=0}^{N-1-(i-k)} s_n(m-i)s_n(m+i-k), \qquad \text{(Eq. 2.27)}$$

Since $r_n(k)$ function is symmetric the Eq 2.21 can be expressed as

$$\boxed{\sum_{k=1}^{p} r_n(|i-k|)a_k = r_n(i)} \qquad 1 \le i \le \text{p} \qquad \text{(Eq. 2.27)}$$

and can be expressed in matrix form as

$$\begin{bmatrix} r_n(0) & r_n(1) & r_n(2) & \cdots & r_n(p-1) \\ r_n(1) & r_n(0) & r_n(1) & \cdots & r_n(p-2) \\ r_n(2) & r_n(1) & r_n(0) & \cdots & r_n(p-3) \\ \vdots & \vdots & \vdots & & \vdots \\ r_n(p-1) & r_n(p-2) & r_n(p-3) & \cdots & r_n(0) \end{bmatrix} \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \hat{a}_3 \\ \vdots \\ \hat{a}_p \end{bmatrix} = \begin{bmatrix} r_n(1) \\ r_n(2) \\ r_n(3) \\ \vdots \\ r_n(p) \end{bmatrix} \qquad \text{(Eq. 2.28)}$$

The $p \times p$ matrix of autocorrelation values is a Toeplitz matrix and can be solved effectively.

2.4.2 Solution of the LPC Equations by using Durbin's recursive solution.

In order to effectively implement a linear predictive analysis system, it is necessary to solve the linear equations. For the autocorrelation method the matrix equation for solving for the predictor coefficients is

$$\sum_{k=1}^{p} r_n(|i-k|)a_k = r_n(i) \qquad 1 \le i \le \text{p} \qquad \text{(Eq. 2.30)}$$

The most efficient method known for solving this particular system of equations is Durbin's recursive procedure which can be stated as follows:

$$E^{(0)} = R(0) \qquad \text{(Eq. 2.31)}$$

$$k_i = \left(R(i) - \sum_{j=1}^{i-1} a_j^{(i-1)} R(i-j) \right) / E^{(i-j)} \qquad 1 \le \text{i} \le \text{p} \qquad \text{(Eq. 2.32)}$$

$$a_i^{(i)} = k_i \qquad \text{(Eq. 2.33)}$$

$$a_j^{(i)} = a_j^{(i-1)} - k_i a_{i-j}^{(i-1)} \qquad \text{(Eq. 2.34)}$$

$$E^{(i)} = (1-k_i^2)E^{(i-1)} \qquad \text{(Eq. 2.35)}$$

Equations 2.32-2.35 are solved recursively for i=1,2,…,p and the final solution is given as

$$a_j = a_j^{(p)} \qquad 1 \le \text{j} \le \text{p} \qquad \text{(Eq. 2.36)}$$

2.5 Filter bank processing

Spectral analysis reveals those speech signal features which are mainly due to the shape of the vocal tract. Spectral features of speech are generally obtained as the output of filter banks. A set of 22-30 band-pass filters is generally used since it simulates human ear processing. Filters are uniformly or non-uniformly spaced along the frequency axis. Usually, the part of the spectrum which is below 1 kHz is processed by more filter-banks since it contains more information on the vocal tract like the first formant. Non-linear frequency analysis is also used to achieve frequency/time resolution.

The most widely used scale in recognition is the Mel scale. The central frequency of each Mel filter bank is uniformly spaced before 1 kHz and it follows a non-uniformly scale after 1 kHz.

A computationally inexpensive method consists of performing filtering in the DFT domain. The DFT response of the m-th filter bank is simply

$$filter_m(f) = \begin{cases} \frac{(filc_{m+1} - f)}{(filc_{m+1} - filc_m)} & if\ f < filc_m\ \&\&\ f > filc_{m-1} \\ \frac{(f - filc_{m-1})}{(filc_m - filc_{m-1})} & if\ f \geq filc_m\ \&\&\ f < filc_{m+1} \\ 0 & else \end{cases} \qquad \text{(Eq. 2.37)}$$

where $filc_m$ is the center frequency of the mth filter bank.

The mth filter bank output is given by:

$$Y_t(m) = \sum_{k=filc_{m-1}}^{filc_{m+1}} X_t(k) \cdot filter_m(k) \qquad \text{(Eq. 2.38)}$$

2.6 Mel frequency cepstrum computation

The final procedure for the Mel frequency cepstrum computation(MFCC) consists of performing the inverse DFT on the logarithm of the magnitude of the filter bank output:

$$y_t(k) = \sum_{m=1}^{M} \log\left\{\left|Y_t(m)\right|\right\} \cdot \cos\left(k\left(m - \frac{1}{2}\right)\frac{\pi}{M}\right), \qquad k = 0, ..., L \qquad \text{(Eq. 2.39)}$$

Since the log power spectrum is real and symmetric then the inverse DFT reduces to a Discrete Cosine Transform(DCT). The DCT has the property to produce highly uncorrelated features. Therefore, in the probability density function of the features, generally modeled by linear combinations of Gaussian functions, diagonal covariance matrices can be used instead of full covariance matrices. This reduces the

computational cost of the number of parameters to be estimated. The zero order MFCC coefficient is approximately equivalent to the log energy of the frame.

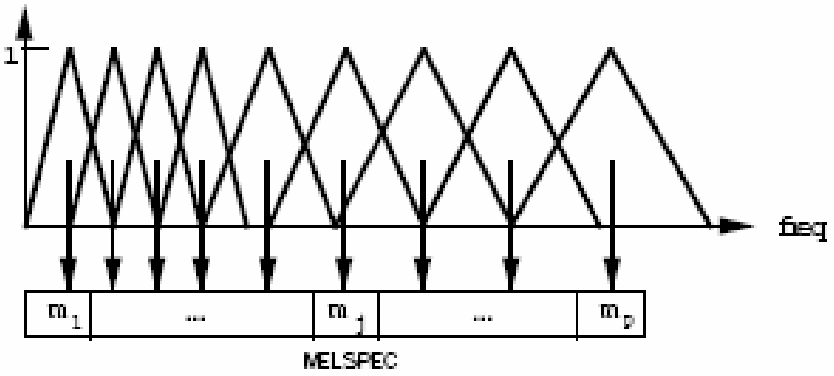

Figure 2. 3 A filter-bank design in which each filter has a triangle bandpass frequency response([6])

2.7 Delta coefficients

A further improvement in performance is obtained by considering that static parameters do not take into account the dynamic evolution of the speech signal, although such evolution carries relevant information for ASRs. First and second order derivatives may be used to capture such information. The delta coefficients are computed using the following formula:

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta (c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \qquad \text{(Eq 2.40)}$$

where d_t is a delta coefficient at time t computed in term of the corresponding static coefficients $C_{t-\Theta}$–$C_{t+\Theta}$. The value of Θ is the number of frames that are at the left or the right side of the static coefficients frame and are used for derivatives computation. Higher order derivative coefficients are computed by applying the same formula. For instance, if we want to compute second order derivatives, we apply the above formula on the delta coefficients.

2.8 Formants

In the context of speech production, the resonance frequencies of the vocal tract tube are called formant frequencies or simply formants. The formant frequencies depend on the shape and dimensions of the vocal tract. Each shape is characterized by a set of formant frequencies. Different sounds are formed by varying the shape of the vocal tract. Thus, the spectral properties of the speech signal vary with time as the vocal tract shape varies,, thus, formants position varies also. Figure 2.4 shows the spectral shape of a frame. Figure 2.5 shows the spectrogram of vowel /aa/ with first four formants clearly marked.

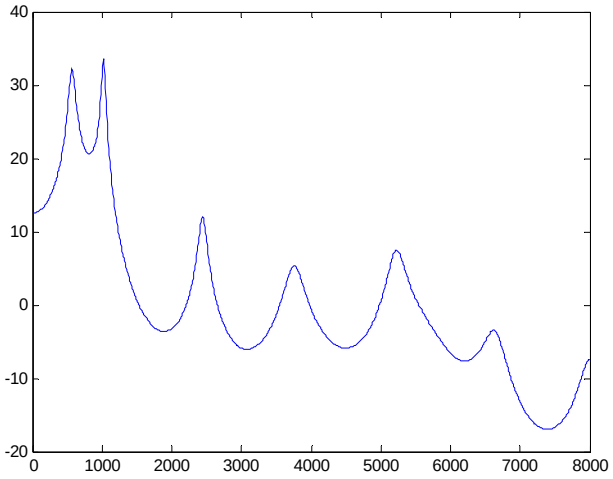

Figure 2. 4 Spectral shape of a frame that has six formants

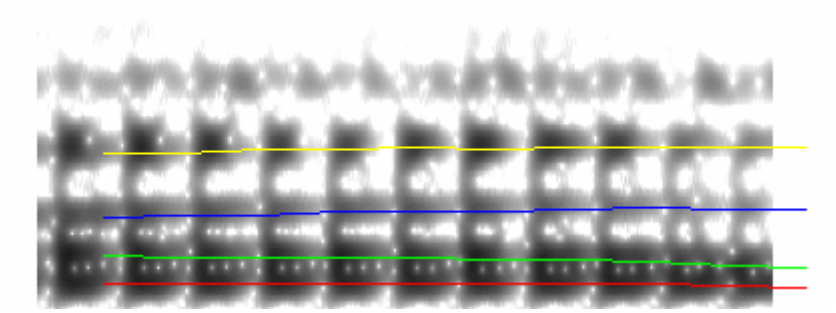

Figure 2. 5 Spectrogram of the vowel /aa/ with first four formants clearly marked

Formants can be estimated from the predictor parameters in one of two ways. The most direct way is to factor the predictor polynomial and try to decide which formants are and which correspond to spectral shaping poles, based on the roots obtained. The alternative way of estimating formants is to obtain the spectrum and choose the formants by a peak picking method.

2.9 Vocal Tract Length Normalization

Vocal Tract Length Normalization (VTLN) is an attempt to improve recognition performance by transforming the system's input. The size of a speaker's vocal tracts is a factor which is directly related to the vocal tract's resonant frequencies (formants). Phone discrimination, relies on observation of the formants, especially on the lowest three (F1, F2 and F3). Small variations in these frequencies are not always salient to human auditory perception but can affect the pattern classification of an automatic speech recognition system.

2.9.1 Maximum Likelihood warp factors

The maximum likelihood method [7] for determining warp factors involves maximizing the likelihoods of speakers' utterances based on a given model usually a HMM. For a speaker I, given a set of models λ along with a set of utterance observation vectors X_i and their transcriptions W_i, the optimal warp factor a_i is defined:

$$\hat{a}_i = \arg\max_a \Pr(X_i^a \mid \lambda, W_i) \qquad \text{(Eq. 2.41)}$$

where a is the factor that is applied to create the warped observations X_i^a. Because equation 2.41 is difficult to solve analytically, the optimal $\hat{a}_i$ is found by a search over a discrete set of factors. The process is illustrated in Figure 2.6

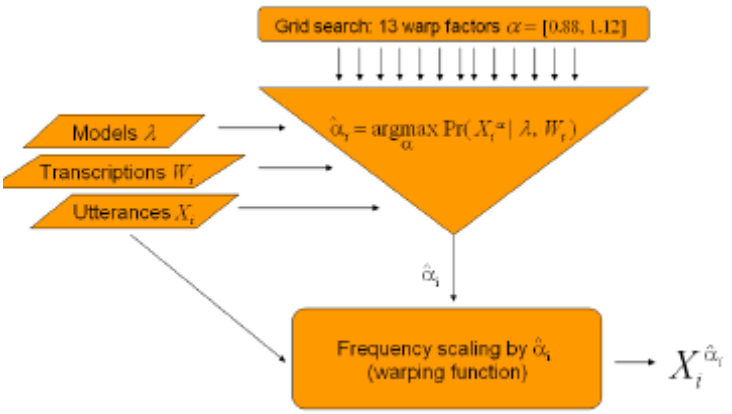

Figure 2. 6 The ML approach to VTLN([7])

The training procedure for ML VTLN consists of a two-pass process. Half of the train set is used for training an HMM while $\hat{a}_i$ is found for the other half and then the sets are swapped. This is iterated until the estimated $\hat{a}^s$ are stabilized between iteration.
The testing procedure involves a multi-pass strategy. In this procedure the optimum $\hat{a}$ is found by aligning warped utterances with respect to a hypothesized word string. Firstly, the unwarped utterance $X_{i,j}$ and the normalized model λ_N are used to obtain a preliminary transcription of the utterance. The transcription obtained from the unwarped features is denoted as $W_{i,j}$. Then, $\hat{a}$ is found by maximizing the likelihood of the warped utterances with respect to the model and transcriptions as follows:

$$\hat{a}_i = \arg\max_a \Pr(X_i^a \mid \lambda_N, W_i) \qquad \text{(Eq. 2.42)}$$

Finally, the utterance $X_{i,j}^a$ is decoded with the model λ_N to obtain the final recognition result.

The primary advantage of the ML approach is that it is design to find a warping factor that is optimal, in a probabilistic sense. However, the process is computationally intensive, especially as the resolution of the warp factor search grid increases.

decoding with every discrete values for a and warping with the $\hat{a}$ that maximizes the probability of the X_i^a given the normalized model λ_N.

2.9.2 Pitch-based Vocal Tract Length Normalization

Pitch-based VTLN [8] exploits the assumption that speakers with large voiceboxes also have large vocal tract; similarly speakers with higher F0 will have smaller larynxes along with shorter vocal tracts and thus higher formant frequencies. Therefore, frequency warping factors can be obtained by speaker's pitch.

The warp factor k as a function of F0

Based on the correlation between pitch and formants, it is possible to derive appropriate warp factors that will transform the formants according to the vocal tract size estimated from pitch. This indicates a warp factor k that is a function of F0:

$$k = f(F_0) \qquad \text{(Eq. 2.43)}$$

In applying VTLN, the warp factor k is the multiplicative constant that accounts for the inter-speaker frequency variations of the formants F and can be used to shift those formants to the normalized F':

$$F' = kF \qquad \text{(Eq. 2.44)}$$

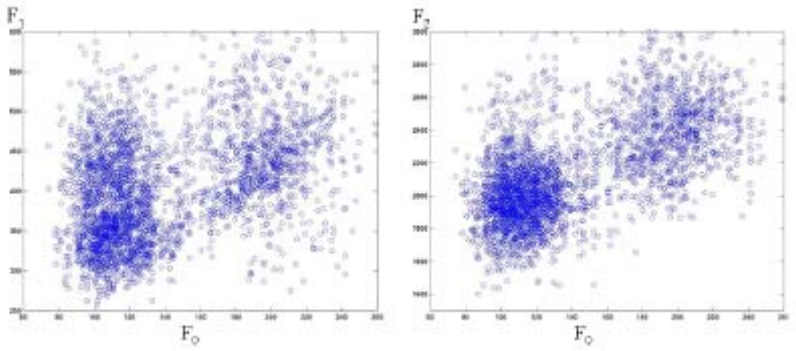

Figure 2. 7 Pitch versus formants F1 and F2 of the vowel /iy/([8])

Given the correlation evident in figure 2.7, it is possible to have a function y that represents the best linear fit, minimizing error in a least-squares sense. This represents the approximate locations of a particular vowel's formants, as a function of F0:

$$F \approx y(F_0) = aF_0 + b \qquad \text{(Eq. 2.45)}$$

To determine the function for the pitch-based warp factor, first consider that the function y has a fixed point at which $y(F_0)$ will be equal to the desired F',

representing the mean frequency to which all other *F* are normalized. The value of F_0 corresponding to this point is μ, the "average" pitch for which the VTLN warp factor equals to 1.

$$F' = y(\mu) \qquad \text{(Eq. 2.46)}$$

Combining Eq 2.44 - 2.46 gives the desired *k* as a function of F_0, along with linear fit coefficients α and b for a particular formant, and the parameter μ:

$$k = f(F_0) = \frac{y(\mu)}{y(F_0)} = \frac{a\mu + b}{aF_0 + b} \qquad \text{(Eq. 2.47)}$$

It is helpful to represent the function's linear approximation to simplify the process of locating the optimal parameters for the Eq. 2.47. This should be done at the value μ to minimize the errors at the extremes of the fundamental frequency domain. Based on linear approximation theory we have that

$$k = f(F_0) \approx L_\mu(F_0) = f(\mu) + f'(\mu)(F_0 - \mu) \qquad \text{(Eq. 2.48)}$$

$$f'(F_0) = \frac{-a(a\mu + b)}{(aF_0 + b)^2} \qquad \text{(Eq. 2.49)}$$

$$L_\mu = 1 - \frac{a}{a\mu + b}(F_0 - \mu) \qquad \text{(Eq. 2.50)}$$

Rewriting the slope of Eq. 2.50 as α, the equation becomes:

$$k = f(F_0) \approx L_\mu(F_0) = 1 - \alpha(F_0 - \mu) \qquad \text{(Eq. 2.51)}$$

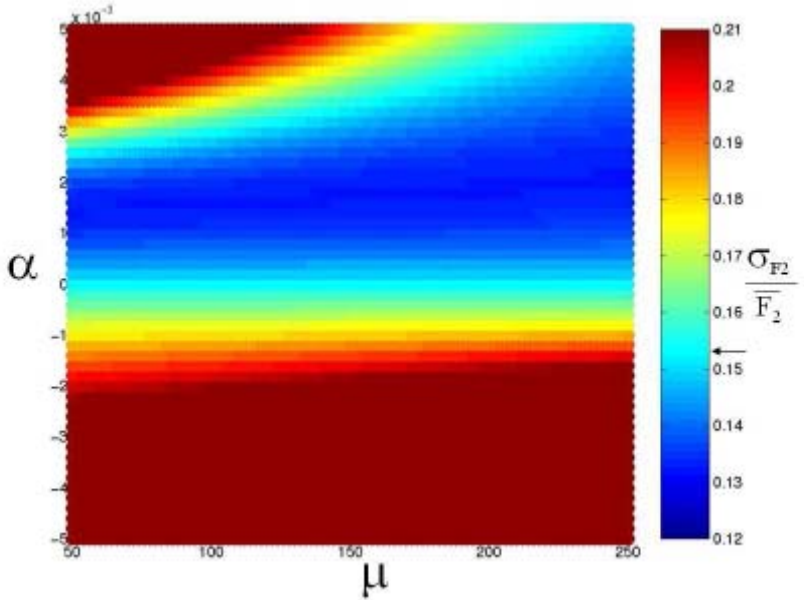

Figure 2. 8 Normalized standard deviation of warped F_2, / iy / ([8])

Based on the above equation we will search for the optimal parameters α and μ of the function f. Figure 2.8 shows the normalized standard deviation of the values of $F_2, / iy /$ when warped by α and μ. It is obvious that an α value of about 0.002 seems to results in the lowest standard deviation of the warped data.

Another measure that we have to consider in the separability of the data is the dicriminability of formants warped by using α and μ. Discriminability of two terms with means m1 and m2 and standard deviation σ1 and σ2 is calculated as follows.

$$d' = \frac{|m_1 - m_2|}{\sqrt{\sigma_1 \sigma_2}} \qquad \text{(Eq. 2.52)}$$

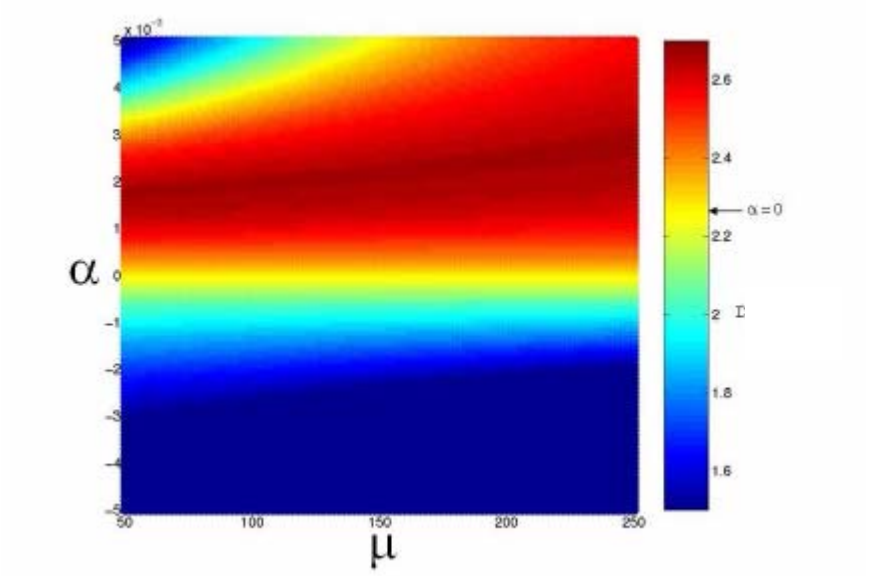

Figure 2. 9 Discriminability of data.([8])

Figure 2.9. clearly show that an α of about 0.002 seems to maximize dicriminability. By examining figures 2.8 and 2.9 we can see that an optimal value for α is 0.002 and a suitable value for μ would be in the range from 130 to 170 Hz. Thus, a suitable approximated formula for the warp factors is:

$$k = 1 - 0.002(F_0 - 150) \qquad \text{(Eq. 2.53)}$$

2.9.3 Warping functions

The most straightforward way to implement VTLN is with a linear warping function. The warped frequencies f' are the product of the warp factor k and the original frequencies f:

$$f' = kf \qquad \text{(Eq. 2.54)}$$

A way to shift frequency is by adjusting the spacing of filters during the feature calculation stage. Figure 2.10 shows the filter-banks that are compressed or expended based on warping factor measure.

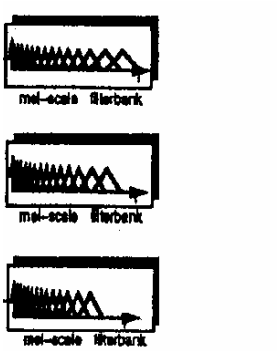

Figure 2. 10 The compression or expansion of the filter-banks, depending on the warping factor([7])

2.10 Summary

In this section, we saw some signal processing techniques that are useful in an ASR system. First, we presented some preprocessing techniques and then some signal processing techniques such as LPC, MFCCs and formant analysis. Finally, we described two vocal tract normalization techniques, the pitch-based and the Maximum Likelihood based VTLN.

3 Classification methods

In this chapter, we will describe two classification techniques that we used in this work. The first technique is based on the nearest neighbor classifier. Moreover, a variant of this technique will be presented, in order to integrate features weighting in the recognition process. We will also see two distance metrics, Euclidean and Mahalanobis distance. The second technique is based on Hidden Markov Models. Finally, an application of HMMs in automatic speech recognition systems will be presented.

3.1 Nearest Neighbor classifier

The k-nearest neighbor algorithm, called k-NN, is a classification method based on closest training samples in the feature space. The training samples are mapped into multidimensional feature space. The space is partitioned into regions by labeling classes of training samples. A point in the space is assigned to a class c if it is the most frequent class label among the k-nearest training samples. Distance metrics that are used usually are Euclidean or Mahalanobis distance.

Firstly, training features vectors are stored and assign their actual class. In the classification phase, testing features vectors are extracted. Also, the distance of the testing vector to all stored vectors is computed and k closest samples are selected. The testing class is predicted to have the most numerous minimum distances within the set.

The choice of k depends on the number of the training samples. Generally, larger values of k reduce the effect of noise on the classification, but make boundaries between classes less distinct. The accuracy of the K-NN algorithm can be degraded by the presence of noisy or irrelevant features. Another problem of the K-NN algorithm is that K-NN is computationally intensive especially when the size of the training set grows.

.
Figure 3.1 shows an example of K-NN decision rule. Samples marked with “+” belong to the first class and samples marked with “x” belong to the second class. The testing point is the round point. Figure shows the Euclidean distance of the k-nearest points in the training space (In this case k=10). Based on 10-NN decision rule, the testing point is classified in the second group because there are eight samples that belong to the second class and two that belong to the first group.

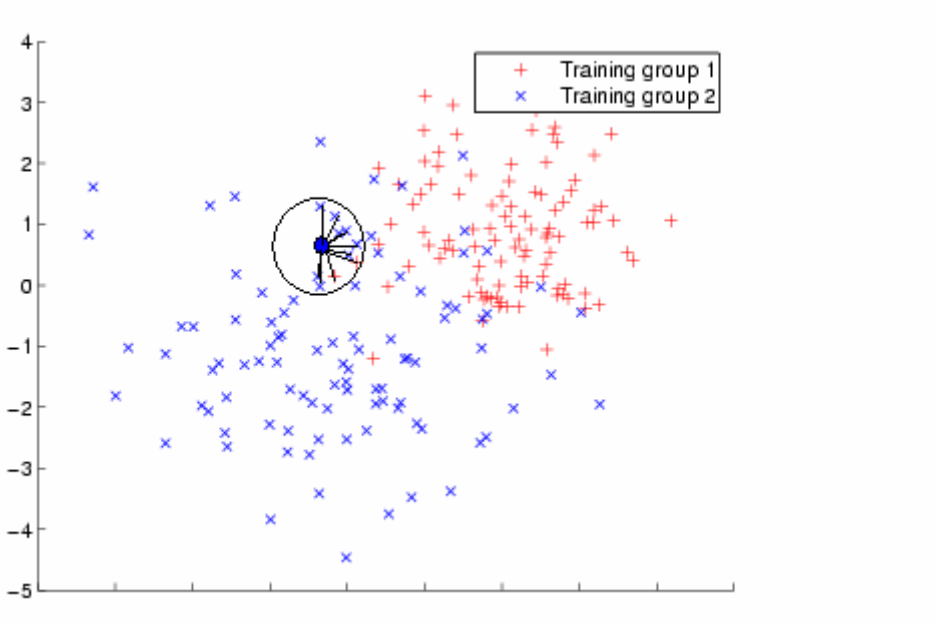

Figure 3.1 The classification by using 10-NN rule.([14])

Another approach to the nearest neighbor rule is the feature weighted K-NN. In this case, each feature that contributes to the distance is weighted by a confidence value c_{ij}. To be consistent, total feature's confidence values for each distance measure must sum up to 1. This leads to the following constrain:

$$\sum_i \sum_j c_{ij} = 1 \qquad \text{(Eq. 3.1)}$$

Feature Weighted K-NN has the advantage of giving more attention to reliable features than unreliable features. Thus, reliable features have greater contribution than unreliable features in the distance calculation. This leads to better classification results in the presence of noise.

3.2 Distance metrics

Distance is a numerical description of how far apart things lie. The most widely used distance metric is the Euclidean distance. The Euclidean distance between two points $P = (p_1, p_2, ..., p_n)$ and $Q = (q_1, q_2, ..., q_n)$, in the Euclidean n-dimensional space, is defined as:

$$\sqrt{(p_1 - q_1) + (p_2 - q_2) + ... + (p_n - q_n)} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \qquad \text{(Eq. 3.2)}$$

The main problem of Euclidean distance as a distance metric is that it depends on the scale of measures.

Mahalanobis distance metric confronts the above defects. Mahalanobis distance is based on correlations between variables by which different patterns can be identified and analyzed. It is a useful way of determining similarity of a features vector to a known feature vector set. It differs from Euclidean distance in that it takes into account the correlations of the data set and is scale-invariant.

Formally, the Mahalanobis distance from a group of values with mean $\mu = (\mu_1, \mu_2, \mu_3, ..., \mu_\pi)$ and covariance matrix $\sum$ for a multivariate vector $x = (x_1, x_2, x_3, ..., x_\pi)$ is defined as:

$$D_M(x) = \sqrt{(x-\mu)^T \sum^{-1} (x-\mu)} \qquad \text{(Eq. 3.3)}$$

3.3 Hidden Markov Models

3.3.1 Markov chains

A first order Markov Chain of N states is a triple (S,A,π), S is a set of N states, A is the $N \times N$ matrix of the transition probabilities between states, π is an N-dimensional row vector of the probability to be in a state at the first time. A Markov Chain property is that the sum of each row of A is one. Another property that holds for Markov Chains is that initial probabilities π_t must sum up to one.

The Markov chain is a state model of a process where one can observe the state transitions that for a first-order Markov Chain depend only on the state at the previous discrete time.

3.3.2 Hidden Markov Models

A Hidden Markov Model is a Markov Chain where output symbols or probabilistic functions describing output symbols are associated to the states. This result to a model with embedded stochastic process with an underlying stochastic process that is not directly observable but can be observed only through another set of stochastic processes that produce the sequence of observations.

An HMM with discrete symbol observations is characterized by:

1. The number of states in the model. Although the states of the model are hidden, there is some significance attached to the states, such as in speech signals. These states are labeled as {1,2,..,N}.

2. The number of distinct observation symbols,M, per state, for example a set of phonemes. These symbols are denoted as V={v1,v2,…,vM}.

3. The state-transition probability distribution A={aij} where

$$a_{ij} = P[_{q_{t+1}} = j \mid q_t = i] \qquad 1 \le i, j \le N \qquad \text{(Eq. 3.4)}$$

4. The observation symbol probability distribution, $B = \{b_j(k)\}$,in which

$$b_j(k) = P[o_t = v_k \mid q_t = j], \quad 1 \le k \le M \qquad \text{(Eq. 3.5)}$$

defines the symbol distribution in state j.

5. The initial distribution in state $\pi = \{\pi_t\}$ in which,

$$\pi_t = P[q_1 = i], \qquad 1 \le i \le M \qquad \text{(Eq. 3.6)}$$

To summarize, HMM specification requires two model parameters N and M, specification of the observation symbols and the specification of the three sets of probability measures A, B and π, described by the compact notation:

$$\lambda = (\mathrm{A}, \mathrm{B}, \pi) \qquad \text{(Eq. 3.7)}$$

3.3.3 Observation densities

The probability to be in each state given an observation X is computed by considering X as a random D-dimensional continuous variable belonging to a suitable pdf. A choice, in this case is to use a multivariate joint density obtained as a product of one-dimensional Gaussian puffs, described by the mean vector μι and by the covariance matrix $\sum_i$:

$$b_t(x) = \frac{1}{\sqrt{(2\pi)^D \det(\sum_i)}} e^{\{-\frac{1}{2}(x-\mu_t)^{\mathrm{T}} \sum_i (x-\mu_t)\}} \qquad \text{j=1,…N} \qquad \text{(Eq. 3.8)}$$

3.3.4 Gaussian mixtures

Unimodal pdfs such as Eq. 3.8 may be unsuitable especially for a multispeaker system. In this case, in fact there are many pronunciation variation of the same

phoneme and the corresponding pdfs are generally multimodal. Therefore, the density shape of Eq. 3.8 could not match the real distribution of speech data. A proper solution is to represent the observations through a mixture of pdfs, for example a linear combination of monomodal densities, where the weight coefficient is c_j^n is proportional to the number of data pertaining to the n-th pdf of the mixture of the i-th state. In essence, a mixture of M pdfs is given by:

$$b_i(x) = \sum_{n=1}^{M} c_i^n b_i^n(x) \text{ with } \sum_{n=1}^{M} c_i^n = 1 \quad , \; c_j^n \le 1 \qquad \text{(Eq. 3.9)}$$

This model allows a better approximation of the observation process even in the presence of articulation effects in the acoustic phonetics variations. The main defect of this model is that the increase of the parameters to estimate requires larger training databases.

3.3.5 An example of a HMM application in ASR.

Let us build a model that recognizes simply the words “yes”, “no” acoustically separated by silence (‘\sil’). Firstly, we require a graphical representation of the acoustic states and a graphical method to associate probabilities. This can be done by using labeled graph where nodes represent states and arc labels are the transition probabilities.

The word “yes” is composed of the two phonemes ‘\ye’ and ‘\s’ corresponding to six states of the two phoneme model (actually the transcription of yes is \y \eh \s), no is composed of two phonemes equal to six states and silence is modeled by one state. Figure 3.2 shows a complete Hidden Markov Model of this simple grammar with a probability density function modeled by a Gaussian mixture

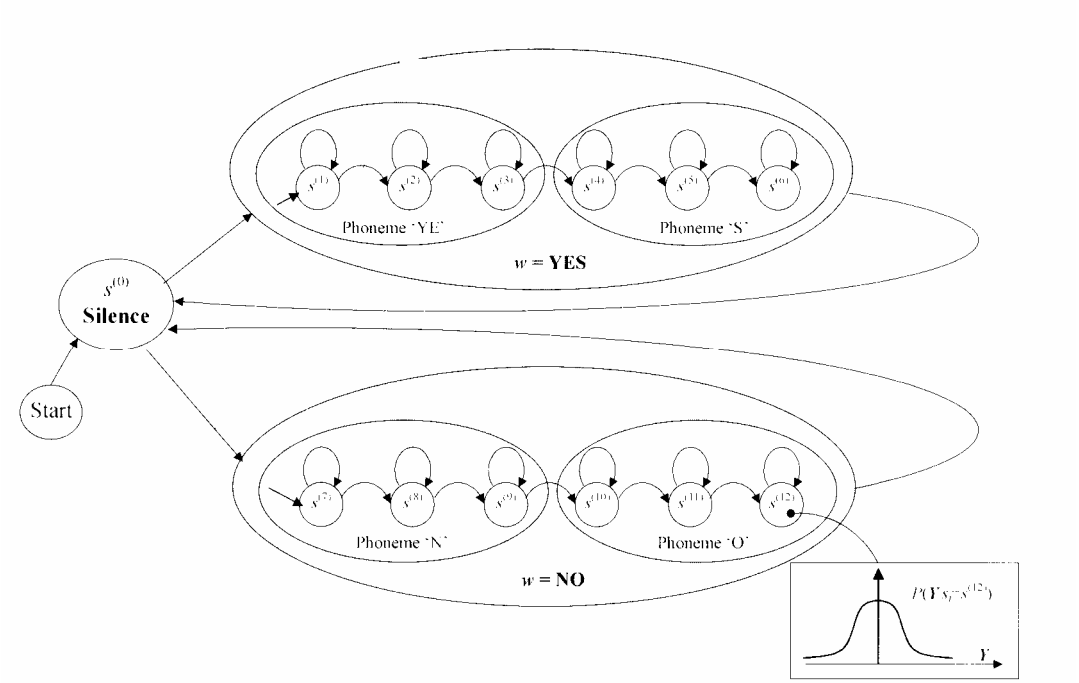

Figure 3. 2 A complete Hidden Markov Model.([1])

3.4 Summary

In this section, we introduced the two classifiers that we used in this work. Firstly, we cited the K–nearest neighbor classifier among with a variant of KNN for feature's weighting during the recognition process. Also, we saw two distance metrics, the Euclidean and the Mahalanobis metrics. Next, a brief explanation of HMMs and of the parameters required to specify an HMM are presented. Finally, we saw an application of HMMs in automatic speech recognition systems.

4 Experimental Procedure

In this section, we will describe the implementation and set-up of the experiments that we did in this work. Firstly, we will present the procedure of extracting our speech database from TIMIT database. Secondly, we will see how we implemented all the signal processing techniques that we used such as formants peak picking, bandwidth estimation and Filter-bank analysis. Lastly, we will describe how we implemented the speech recognition systems that we used in order to classify speech frames and phonemes.

4.1 Training and testing phonemes extraction.

In order to extract phonemes, we created a script that automatically extracts phonemes from TIMIT database and stores them in raw format. The following phonemes were extracted and used in all experiments:

{ ‘aa’ , ‘ah’ , ‘ax’ , ‘eh’ , ‘ih’ , ‘eh’ , ‘ow’ , ‘ux’ }

For each phoneme, we randomly select a dialect and a speaker. Then, we randomly select an utterance, which is spoken by the particular speaker, that includes the specific phoneme and finally I save the phoneme in raw format. After determining the bounds of the phoneme in the signal, we do zero padding (append one or more zeros to the end of a signal) in order to make the signal length an integer multiple of the window length.

For K-NN experiments, we extracted 300 unique phonemes for training and 200 unique phonemes for testing (totaling 2400 for training and 1600 for testing). For HMM-based experiments, we extracted 900 unique phonemes for training and 560 unique phonemes for testing (totaling 7200 for training and 4480 for testing). The HTK experiments have more testing and training phonemes because HTK-based recognition is more computationally efficient than NN-based recognition. We used the database which includes 2400 training and 1600 testing phonemes for all KNN experiments and we used the database which includes 7200 training and 4480 testing phonemes for all HTK experiments.

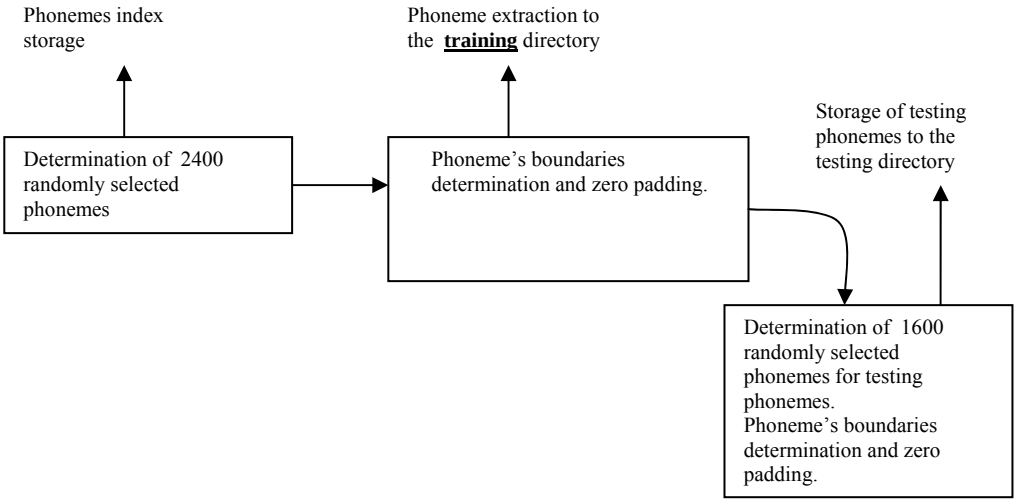

Figure 4. 1 Phoneme extraction procedure from TIMIT database

4.2 Preemphasis

As discussed in Chapter 2, it is helpful to use a preemphasis filter. We use the following filter on the speech signal during the pre-processing process.

$$H(z) = 1 - 0.95 * z^{-1} \qquad \text{(Eq. 4.1)}$$

The usage of this filter aims to emphasize high frequencies that have low amplitude values. The figure 4.2 shows the spectrum of the speech signal before applying the preemphasis filter and the figure 4.3 shows the spectrum of the signal after the application of the preemphasis filter.

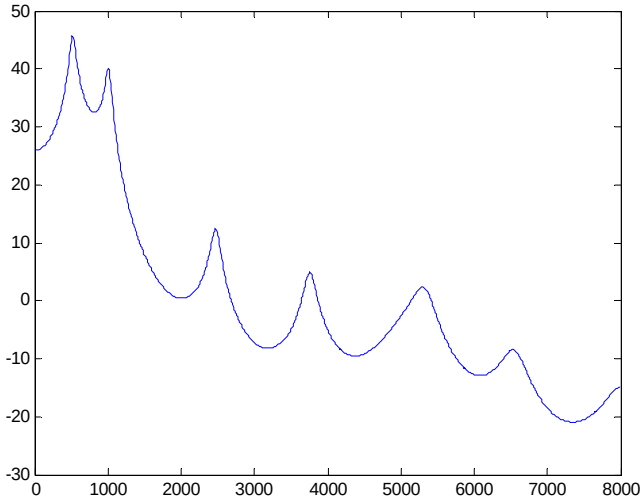

Figure 4. 2 Spectrum of the signal without applying the preemphasis filter

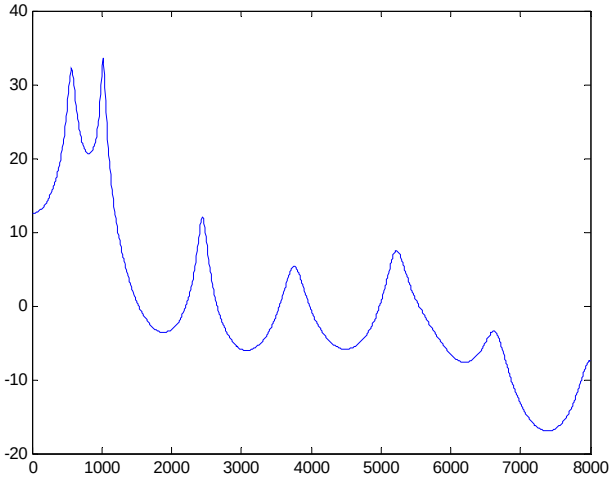

Figure 4. 3 Spectrum of the signal with preemphasis application

4.3 Windowing

After applying the preemphasis filter, the signal is windowed by applying on it a hamming window. By windowing the signal, we can process the speech signal as a stationary one and extract its spectral characteristics. For this purpose, we use a window that has length 320 samples (20ms duration on 16Khz sampling frequency). This means that a speech frame has length 320 samples.

To extract the spectral features of the whole signal, we shift the window by 160 samples (10 ms). Figure 4.4 shows the Hamming window that is applied on the speech signal and Figure 4.5 represents the speech signal in the time domain after the application of the window on it.

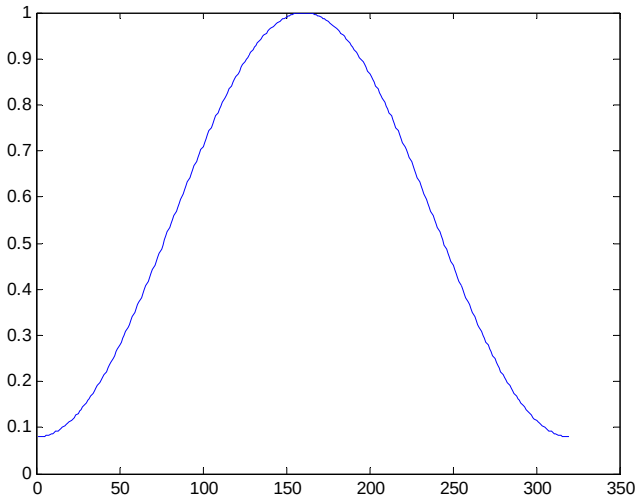

Figure 4. 4 20ms Hamming window

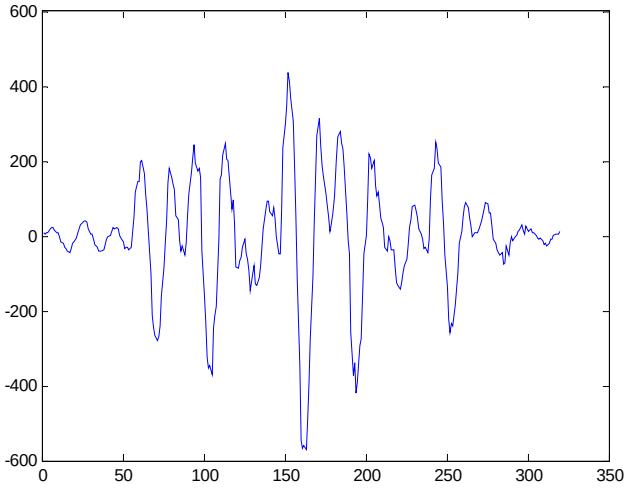

Figure 4. 5 The windowed signal

4.4 Peak picking method

In order to use formants as features, we have to use a method to estimate their frequencies and their bandwidths. An efficient method for estimating formant's frequencies is the peak picking method. Firstly, we extract LPC coefficients from the signal. Based on the vocal tract system that is described by LPC coefficients, we use Fast Fourier transform to extract the spectral shape of the windowed signal.

After the spectrum analysis of the windowed signal, we have to estimate formant frequencies and their bandwidths. In figure 4.6, we have plotted the spectrum of the speech signal shown in figure 4.5. It is clear that there are 6 formants. The first two formants have higher amplitude and lower bandwidths than higher order formants.

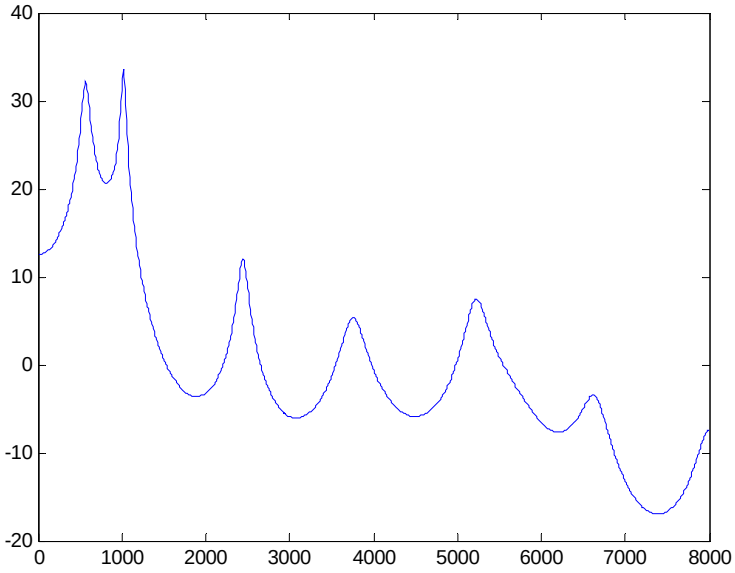

Figure 4. 6 Spectral shape of the windowed signal

By applying pick picking method on the spectral shape of the signal, formant frequencies are picked in sequence, by picking all local maxima of the cross-section spectrum. Then, we estimate bandwidths of those local maxima by finding high and low frequencies with amplitude 3dB below the particular maximum. If there is no high or low frequency with amplitude 3 dB below the particular maximum then we estimate the frequency by expanding the imaginary curve of the formant. By subtracting the high frequency from the low frequency, we estimate the bandwidth of the formant. Figure 4.6 shows the frequency and bandwidth estimation of the 5th formant. To calculate the bandwidth of the 7th formant we have to predict the low3db by expanding the curve.

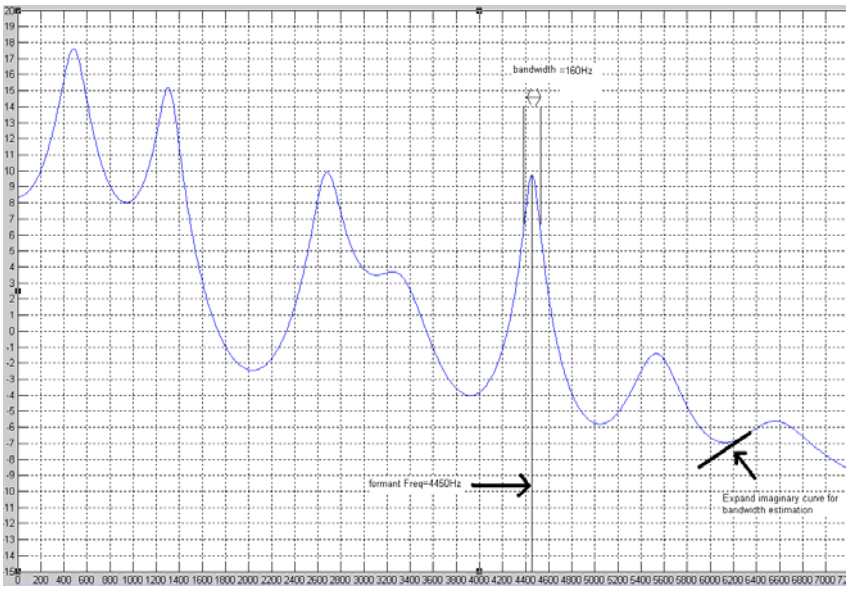

Figure 4. 7 Frequency and bandwidth estimation.

4.5 Vocal Tract Length Normalization

In this section, we will describe two approaches implemented for normalizing the vocal tract length. The first approach is the pitch-based approach and the second is the Maximum Likelihood approach to extract warping factors.

For the pitch-based approach, training warp factors were calculated for each speaker by using the warp factor formula which is described in chapter 2 and is using the speaker's average pitch. The pitch was detected by a reliable pitch tracking tool (Yin tool) and the average pitch for a speaker was taken to be the mean of the median over all speaker's utterances. Testing warp factors were estimated by the median pitch of the utterance that contains the particular phoneme. The pitch contour that was extracted by using the wavesurfer tool is shown in figure 4.7.

Figure 4. 8 Pitch contour of a complete utterance.

Maximum Likelihood warp factors were calculated while training Gaussian Mixture Models with formant coefficients. However, in this process we estimated warping factors on the phoneme models and not on word models that could give better results.

The procedure that I used to warp formant frequencies and bandwidth is simple. We just multiply the frequency by the warping factor that is extracted by using the above to approaches.

4.6 The MFCCs extraction process

In order to compare recognition results of our approach with the classic recognition method that uses MFCCs as features, we have to compute MFCCs. MFCCs are extracted by using the MFCCs method described in chapter 2. We used 30 filter banks which are uniformly scaled on frequencies less than 1 kHz. The filter-banks positioned above 1 kHz, have bandwidth which is increasing by a factor of 1.1 as we increase the number of filters. Figure 4.8 shows the filter-banks used for the MFCCs extraction process.

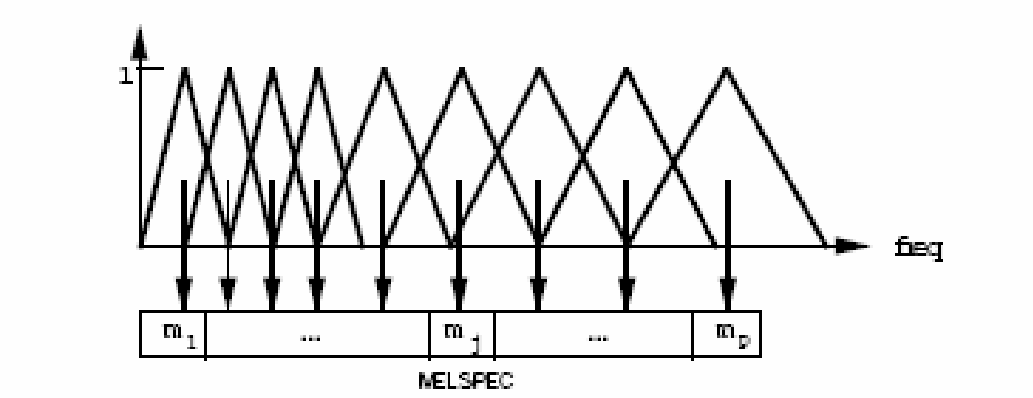

Figure 4. 9 A filter-bank design used for MFCC extraction([6])

In the case of VTLN, filter-banks bandwidth is scaled by a factor, α, which is computed in the VTLN procedure. VTLN causes filter-banks to expand or compress, depending on the warping factor.

4.7 Delta coefficients calculation

To be able to extract derivative features, some pre-processing has to be made. Firstly, I have to locate the signal that precedes or follows each phoneme. After locating this signal, those frames are stored in the same way that phonemes were stored. The number of extra frames that are stored is proportional to the number of extra frames needed to calculate first and last feature's derivatives. The same procedure takes place for the calculation of the second derivative. The storage structure of all extra frames is the same as the storage structure of phonemes.

After extracting all required extra frames, we can extract the derivatives of the static features. Depending on the derivative that we want to calculate (first or second derivative) we append the appropriate number of frames to the signal. For example, if we use two frames that precede the frame and two frames that follow the frame for the first derivative calculation then we have to add four extra frames to the database. Then, we can differentiate the signal by using the formula

$$d_t = \frac{\sum_{\theta=1}^{\Theta} \theta(c_{t+\theta} - c_{t-\theta})}{2\sum_{\theta=1}^{\Theta} \theta^2} \qquad \text{(Eq 4.2)}$$

where θ is the number of left or right frames that are involved in this procedure.

4.8 Confidence measures

The main defect of formants is that they are affected by noise and this leads to disastrous results in formants based speech recognition. To solve this problem we introduce some confidence measures that we will use to weight feature's distance. This is based on the idea that we can weight more features that are robust during the speech recognition process.

The first confidence measure that we used was based on the curvature[12] and the amplitude of the peak. To compute this confidence measure we specify each spectral section as a vector

$$S = s[1],...,s[1024]$$

where one unit on the scale corresponds to 0.25 dB.

Before the confidence calculation we have to calculate two weight measures of each peak, named w1 and w2.

$$w_1 = (S[f] - M_{amp} + W_{shift}) / W_{scale} \quad \text{(Eq 4.3)}$$

$$w_2 = (S[f] - REF_{LEV} + REF_{Wshift}) / REF_{Wscale} \quad \text{(Eq 4.4)}$$

The above weights are based on the relative value of the formant amplitude, the maximum amplitude on the current spectral section (w1) and the long-term maximum amplitude (w2).

where

$$M_{amp} = \max(S[i]), where \quad i = 1,..,1024 \quad \text{(Eq 4.5)}$$

REFLEV is the maximum spectral channel amplitude

and the empirical constants

WTshift=100
WTscale=4
REFWshift=140
REFWscale=5

The total weight is the minimum weight between w1 and w2 and if the total weight is higher than 16 it is set to 16.
Also, confidence depends on curvature and it is estimated as follows

$$C = 2 * s[f] - s[lo] - s[hi], where \ lo = f - 64 \quad and \quad hi = f + 64 \quad \text{(Eq 4.6)}$$

If curvature is greater than 32,it is set to 32 and if it is lower than 8 it is set to 8.
Finally, confidence is calculated by using weight and curvature as follows:

$$conf = w * C / 512 \quad for \ \ w * C > 0$$
$$conf = 0 \quad for \ \ w * C \le 0 \quad \text{(Eq 4.7)}$$

Two other confidence measures are based on formants bandwidth and formants relative height (shown in figure 4.10). To calculate the bandwidth-based confidence, we suppose that bandwidth and confidence are connected with the following formula:

$$conf = \sqrt{2\pi} * 200 * \mathrm{N}(0, 200) \quad \text{(Eq 4.8)}$$

where N(0,200) is a Gaussian distribution with zero mean value and standard deviation 200.

To calculate the height-based confidence, we suppose that height and confidence are connected with the following formula:

$$conf = \sqrt{2\pi} * 15 * \mathrm{N}(0,15) \qquad \text{(Eq 4.9)}$$

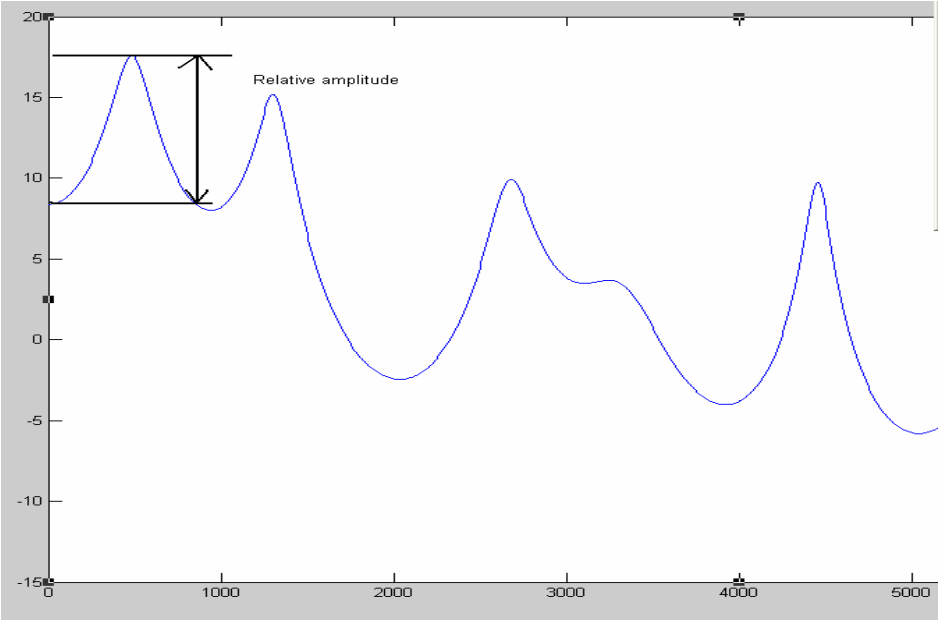

Figure 4. 10 The relative amplitude calculation of the first formant

The last confidence measure that we used was the bandwidth-based confidence measure using a cut-off frequency. In this type of confidence measure, we weight each feature according to the following formula.

$$conf = \begin{Bmatrix} 1 \; bandwidth < 1000 \\ 0 \; else \end{Bmatrix} \qquad \text{(Eq 4.10)}$$

4.9 Classifiers set-up and implementation

In this section, we will describe the classifiers used for recognition. Generally, we used two classifiers, the K-NN-based classifier and the HTK-based classifier. K-NN is used for frame recognition and HTK is used for phoneme recognition. As discussed above, the HTK database experiments database is bigger than K-NN database because HTK-based recognition is more computationally efficient that K-NN based recognition.

4.9.1 K-NN-based classification

The first classifier that we used was the NN-R classifier. This classifier does not require a training process and as described in theory we search for the k-nearest training samples of the testing vector. This gives us the flexibility to incorporate

confidence measures not only from the testing vector but also from the training vector.

The first experiment that we tried was to use the frame's first three formants as features. Firstly, all training vectors, consisting of first three formants was created and saved. After the extraction procedure, we calculated the variance of all training vectors in order to normalize the distance in the classification procedure. Then, in the classification procedure, we extracted all testing vectors and for each testing frame we calculated the mahalanobis distance from all training vectors and saved in the first array the 25th closest distances from the testing frame among with the class labels of the training vectors. Lastly, we searched for the class with the most minimum distances for each case. This class is marked as the class of the testing vector. We repeated this experiment by storing the 50th closest distance from the testing frame.

Many variants of the above experiment were done. Each time we changed the features for recognition. The experiment was done by using 3 MFCCs , 6 MFCCs ,12 MFCCs, 12 MFCCs and 3 three formants ,3 formants and their corresponding bandwidths. Also, the above experiments were repeated by using vocal tract normalization techniques during the extraction process.

Another set of K-NN experiments was to incorporate confidence measures during the classification process by using information from both the testing and training vector. We did experiments by using confidence measures based on the curvature, on the bandwidth of the formant and on the height of the formant. Also, in order to combine the confidence measures of the testing and the training vector we tried two combination methods. The first method was to combine them by multiplying the confidence measures of the training and the testing vector and the second method was to combine them by summation.

Moreover, we tried to estimate the recognition accuracy in the case of including derivatives of the static features in the features set. So, we estimated the first and the second derivative of MFCCs and the first three formants. We included the above dynamic features in experiments with 12 MFCCs and in experiments with 3 formants.

Furthermore, we tried to include all formants in the recognition process. By incorporating all formants, we faced this problem: In some frames there were missing formants and others had more formants due to noise. To overcome this problem during the distance calculation, we spotted the vector that had the minimum number of formants and we shifted it in order to calculate all possible distances. Each time, the distance between the two frequencies was multiplied by a weighting measure resulting from a combination of confidence values based on formants position. This type of confidence measures was a sigmoid function calculated by the formula:

$$FormantPositionConfidence = \frac{1}{1+e^{\frac{100}{5*range_{\max}-range_{\min}}(-x+range_{\min})}} - \frac{1}{1+e^{\frac{100}{5*range_{\max}-range_{\min}}(-x+range_{\max})}}$$

(Eq 4.11)

where the range for each formant is given by the following table:

Formant	Rangemin	Rangemax
F1	150	900
F2	500	2500
F3	1300	3500
F4	2500	4500
F5	3500	4900
F6	4000	6000
F7	5500	7000
F8	6000	8000

An example of this experiment is given below:

Let's say that the first frame has 3 formants and the second frame has 2 formants. We calculate the distances as shown on the following diagram. We combine all possible formants alignments and I calculate their distance. To find the total confidence, we combine Sigmoid Confidence values and bandwidth-based values of each formant and we normalized the distance of all formants to sum up to N. Then, I multiply normalized total confidence with its corresponding distance and I calculate their summation. Finally, I apply the K-NN rule to decide the class of the frame.

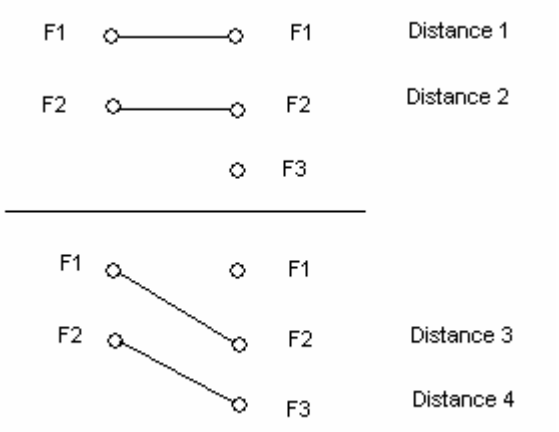

Figure 4. 11 Distance estimation example

As said above, to be consistent, total confidence must sum up to N for every frame.

$$\sum_i \sum_j c_{ij}' = N \qquad \text{(Eq 4.12)}$$

where N is the total number of formants that I combine in each frame's total distance.

To achieve this I multiply each confidence value of a frame with $\frac{N}{\sum_i \sum_j c_{ij}}$

so the confidence becomes

$$c_{ij}' = \frac{N}{\sum_i \sum_j c_{ij}} c_{ij} \qquad \text{(Eq 4.13)}$$

4.9.2 HMM-based classification

The second type of experiments done was the experiments based on HMM modeling. To do so, we used the HTK Toolkit, which is an open source tool. Due to the properties of HMMs, we can use HTK to recognize phonemes and not only single frames as done in K-NN experiments.

In order to use features extracted in the front-end, we have to save them in a format that is readable by the HTK. An HTK feature file is consisted of a continuous sequence of samples preceded by a header. Each sample is a vector of either 2-byte integers or 4-byte floats. The HTK file format header is 12 bytes long and contains the number of samples which is a 4-byte integer, the sample period in 100ns units which is a 4-byte integer, the number of bytes per sample which is a 2-byte integer and finally a code indicating the sample kind which is 2-byte integer. The first parameter and the third parameter vary depending on the number of frames of each phoneme. The second parameter is set to 200000 which correspond to 20ms. The fourth parameter is always set to 9 which mean that we have a user defined feature type.

In contrast with K-NN, HTK needs a training procedure before starting the recognition process. Before training the HMM model, we have to create an HMM protocol file which actually explains to HTK the parameter of the HMM network. In all experiments, we used three emitting states and diagonal variance. Also, we did experiments with 1,3,6 and 8 Gaussian mixtures. An HMM prototype is shown in figure 4,12. In order to train all phonemes, we used this prototype to initialize and train all phonemes HMMs.

```
~h "proto"
<BeginHMM>
<VecSize> 3 <USER>
   <NumStates> 5
   <State> 2 <NumMixes> 1
     <Mixture> 1 1
       <Mean> 3
         0.0 0.0 0.0
       <Variance> 3
         1.0 1.0 1.0
   <State> 3 <NumMixes> 1
     <Mixture> 1 1
       <Mean> 3
         0.0 0.0 0.0
       <Variance> 3
         1.0 1.0 1.0
   <State> 4 <NumMixes> 1
     <Mixture> 1 1
       <Mean> 3
         0.0 0.0 0.0
       <Variance> 3
         1.0 1.0 1.0
   <TransP> 5
      0.0 1.0 0.0 0.0 0.0
      0.0 0.5 0.5 0.0 0.0
      0.0 0.0 0.5 0.5 0.0
      0.0 0.0 0.0 0.5 0.5
      0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Figure 4. 12 An HMM prototype

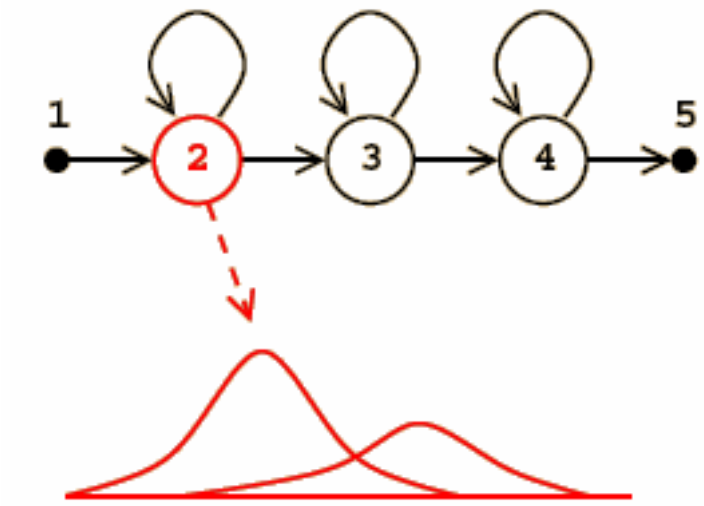

Figure 4. 13 The structure of the HMM prototype

After training all phonemes apart by using the above procedure, we concatenate all HMMs files that come up from the previous procedure into one file, in order to make the final training including all HMMs.

Before starting the recognition process we have to create the dictionary and the network that will be used in the recognition. The dictionary is fairly simple and it includes only the eight phonemes that we want recognize. Figure 4.14 shows the dictionary used in recognition.

```
aa [aa] aa
ah [ah] ah
ax [ax] ax
eh [eh] eh
ih [ih] ih
ix [ix] ix
ow [ow] ow
ux [ux] ux
```

Figure 4. 14 phoneme dictionary

The structure of the network is shown in figure 4.15

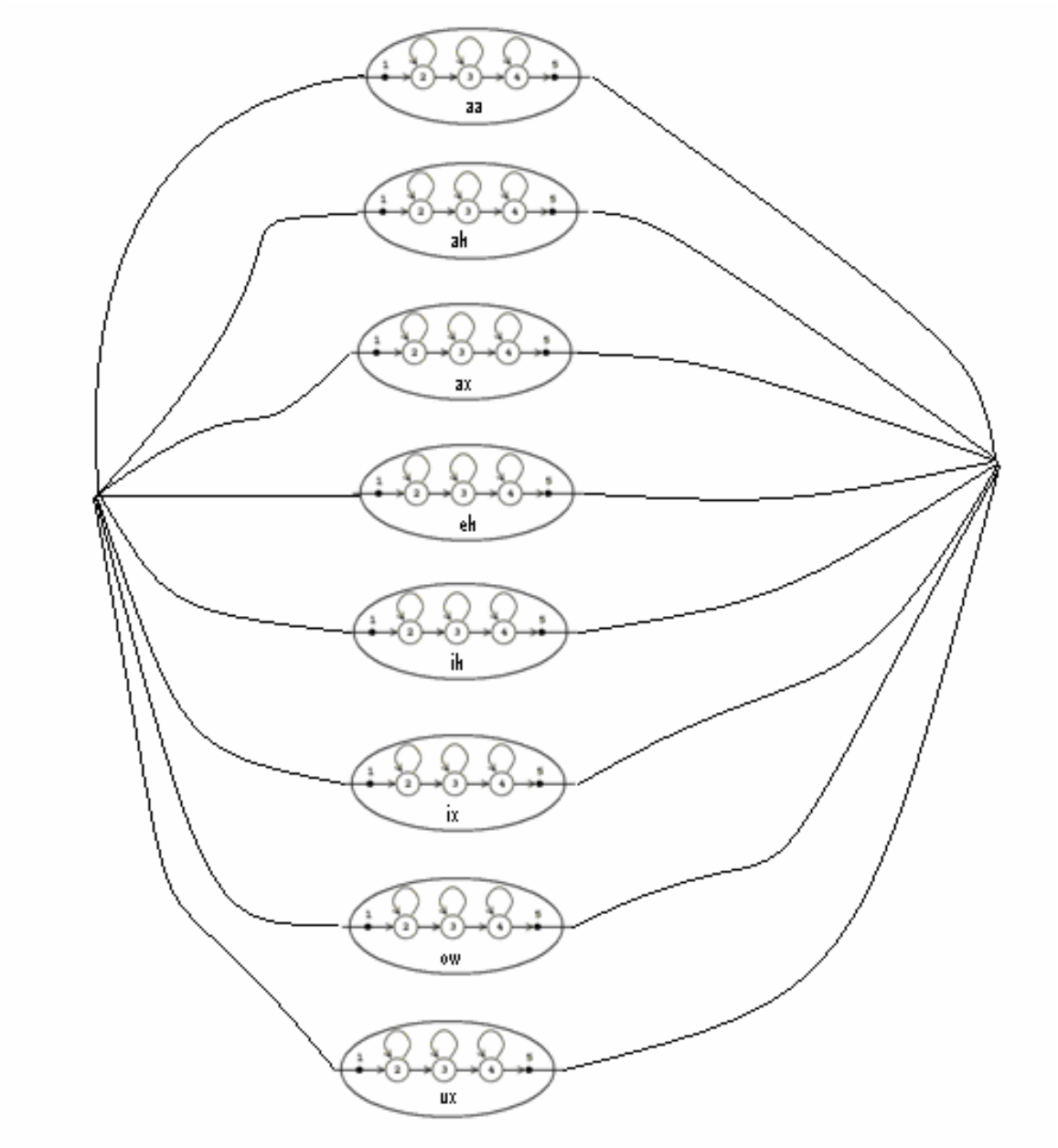

Figure 4. 15 Structure of the network

The last stage in this process is to recognize phonemes. In order to do this, we used the above trained models, the phoneme dictionary and the network with the structure shown above. After phonemes recognition, we incorporate testing phonemes transcriptions for extracting recognition accuracy in a human readable form. Figure 4.16 shows the complete recognition process, from the feature extraction process through to recognition.

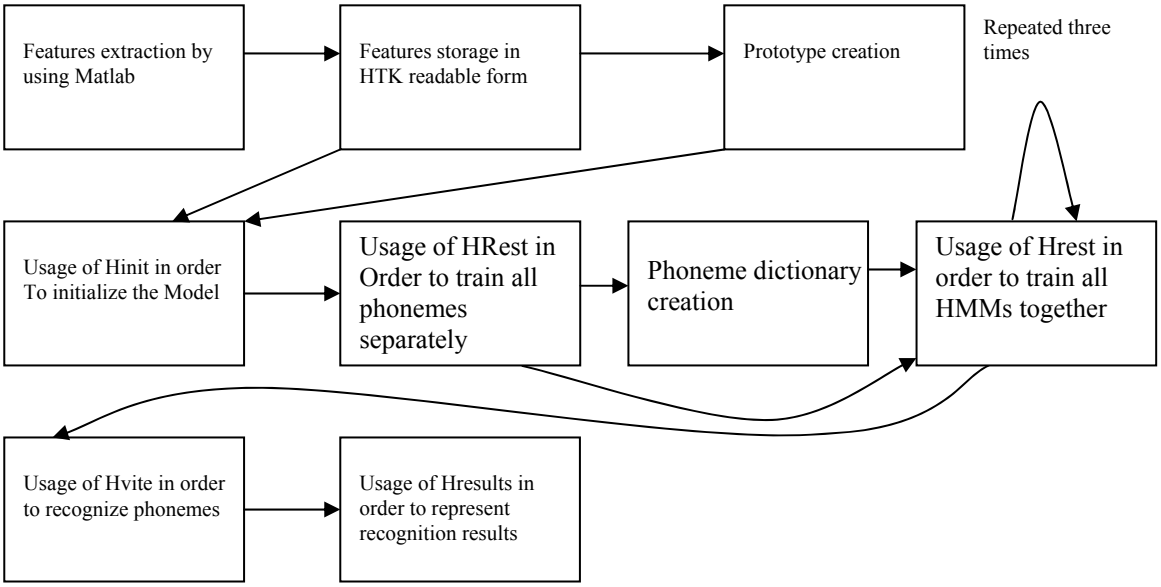

Figure 4. 16 The complete recognition process by using HTK

We did the above experiment using as features 3 MFCCs, 6 MFCCs, 12 MFCCs, 3 formants and 12MFCCs among with 3 MFCCs. Also, all the above experiments were done by applying vocal tract normalization during the feature extraction process. All the above experiments were repeated by using one and three Gaussian mixtures, in order to compare the accuracy coming from a different number of mixtures.

Another set of experiments done was by including first and second derivatives as features in the above process. This experiment was done only for the cases of 12 MFCCs and 3 Formants extracted by applying vocal tract normalization.

Furthermore, we tried to include the confidence measures described above. To do so, we modified the HTK source code to be able to read weights from files having the same format as the features files. Also, we had to alter the mahalanobis distance calculation process for the single and multi-mixture cases in order to integrate weights given by the confidence file. Based on this modification, we did some experiments with confidence measures extracted by using curvature, bandwidth and height.

4.10 Summary

In this chapter, we saw the procedure that we used to extract our speech database from the TIMIT database. Moreover, we described how we implemented features extraction techniques such as peak picking, MFCCs analysis and vocal tract normalization. Finally, we described all the recognition systems that we implemented and set-up, including K-NN and HTK among with some variants.

5 Evaluation Procedure

In our work, in order to evaluate the recognition quality and compare various recognizers, we have to estimate their accuracy for each experiment done. However, the accuracy metric used for K-NN experiments differs from the accuracy metric used in HTK experiments. The reason that this happens is because in HTK experiments we evaluate the total amount of correct phonemes and in K-NN experiments we evaluate the total amount of correct frames.

5.1 Accuracy

Firstly, we will present how overall accuracy is estimated for K-NN experiments. By saying overall accuracy in K-NN experiments we refer to the percentage of the total correct frames. In order to estimate the accuracy percentage for K-NN experiments we use the following formula:

$$accuracy = \frac{total \quad number \quad of \quad correct \quad frames}{total \quad number \quad of \quad frames} \times 100\% \qquad \text{(Eq. 5.1)}$$

As said above, we cannot use the average accuracy of all phonemes because each phoneme has a different number of frames leading to wrong accuracy results, so during the recognition procedure, we count the number of correct frames and the number of total frames.

In HTK experiments, accuracy[6] is calculated as follows:

$$Percentatge \ \ accurancy = \frac{N - D - S - I}{N} \times 100\% \qquad \text{(Eq. 5.2)}$$

where N is the total number of phonemes, S is the number of substitution errors, D is the number of deletion errors and I is the number of insertion errors. In the case of phoneme classification, we have only substitution errors so Eq 5.2 becomes:

$$Percentatge \ \ accurancy = \frac{N - S}{N} \times 100\% \qquad \text{(Eq. 5.3)}$$

The accuracy tells us how well a recognizer classifies phonemes or in other words they provide us a good mathematical way to compare the classification performance of our experiments.

5.2 K-NN-based Experiments

5.2.1 Recognition performance of various combinations of features

The first set of K-NN experiments that we tried, aimed to compare the recognition performance of the classifier by using different combinations of features. The features used in this set of experiments were 3 formants, 3 MFCCs, 3 formants and their bandwidths, 6 MFCCs, 12 MFCCs and 12 MFCCs along with 13 formants.

The above experiments were done by using both 25-NN and 50-NN classifiers. Results of the above experiments are presented graphically in figure 5.1.

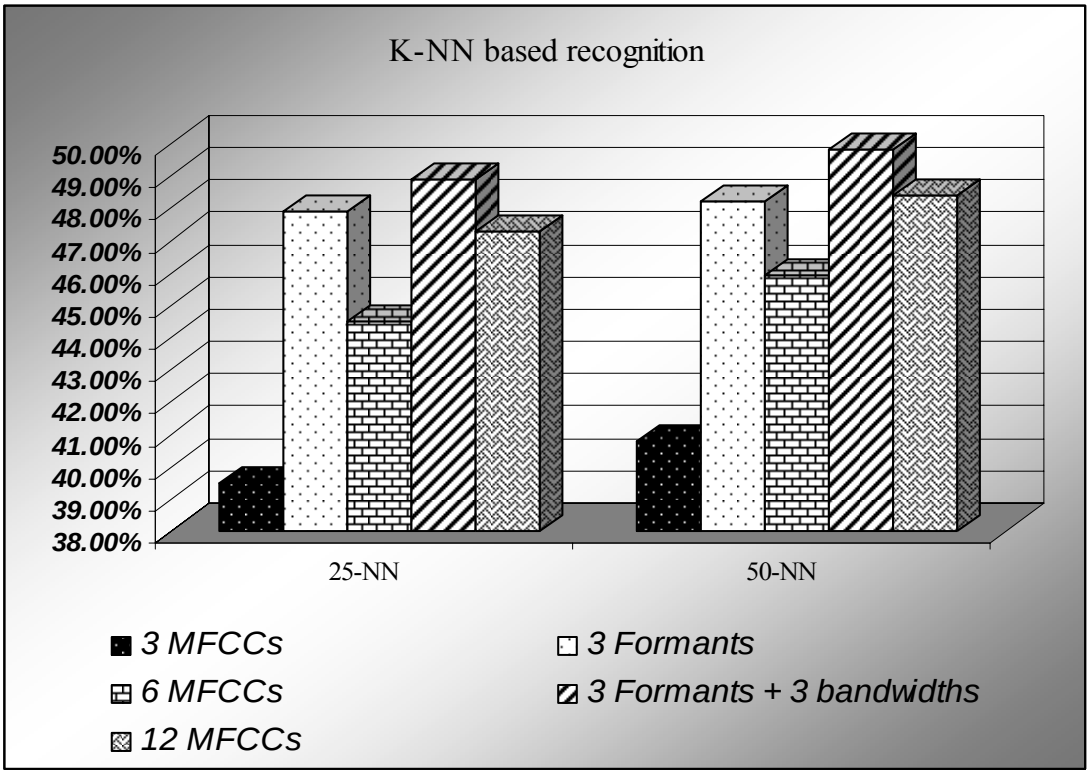

Figure 5. 1 The recognition performance of various combinations of features

The above results show that when comparing the same number of features, formant-based recognition outperforms MFCC-based recognition, giving ~8% higher recognition accuracy. Experiments with three formants perform fairly well and their results are slightly lower than the results achieved by using the 50-NN classifier along with three formants and their bandwidths as features. However, this small difference might be due to a statistical error. Generally, the above results show that we can achieve the performance of twelve MFCCs with a lower number of features.

5.2.2 Recognition performance of vocal tract length normalization

Furthermore, we examined the effect of VTLN in the case of frame classification and the effect of appending bandwidths in the feature set. In the first experiment, the feature vector is consisted of the first three formants and in the second experiment the feature vector is consisted of the first three formants and their bandwidths, totaling six features.

Both experiments were repeated by applying the pitch-based VTLN during the features extraction procedure. Also, the above experiments were done for both 25-NN and 50-NN classifiers. Results are presented graphically in figure 5.2.

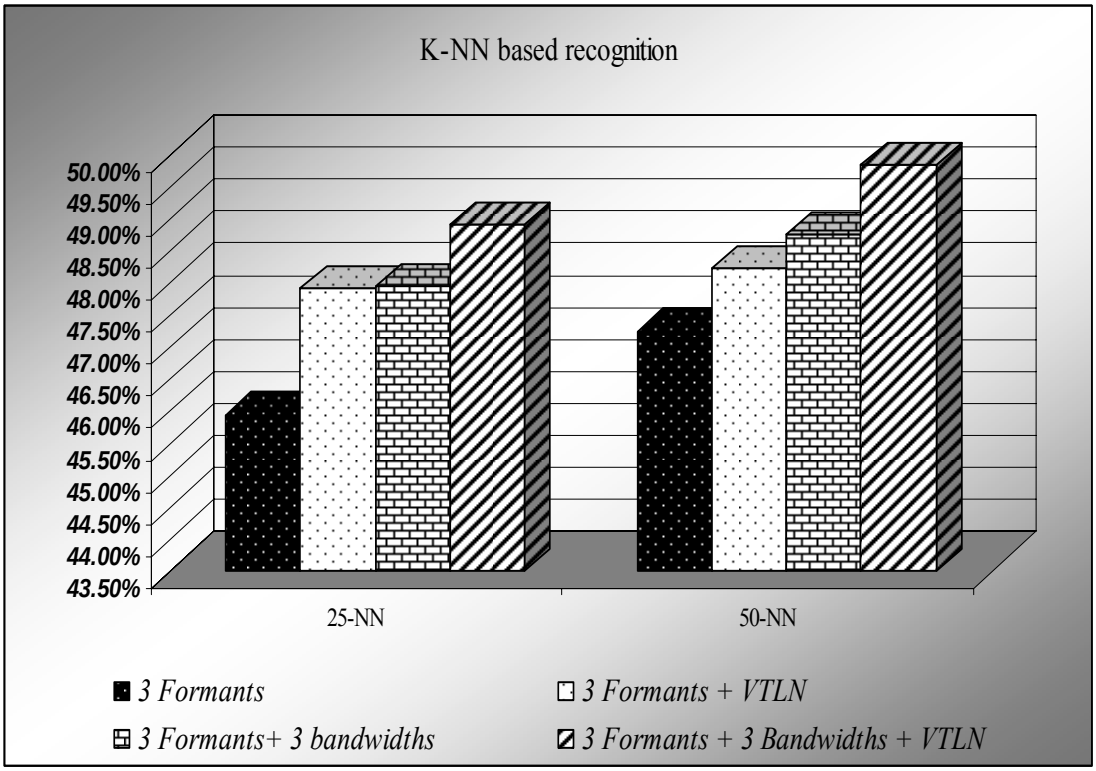

Figure 5. 2 The recognition performance of the VTLN application

Obviously, the results of this experiment show that the application of pitch-based VTLN is improving the accuracy. It is remarkable that when bandwidths are added to the feature set, results are improved slightly but this might also be caused by a statistical error. Also, the highest accuracy is achieved by using the first three formants and their bandwidths as features and by applying VTLN during the extraction process.

5.2.3 Performance of formants-based recognition by using confidence measures

The purpose of this set of experiments is to improve recognition results of recognizers which used the first 3 formants as features. As said in theory, formants are highly influenced by noise. In this experiment, we tried to weight clean features more than

those affected by noise. Possibly, formants with high peaks and low bandwidth are cleaner than those with larger bandwidths. Based on this notion, we tried to combine confidence measures of training and testing vectors, in order to improve recognition results.

In all cases, the feature vector consists of the first 3 formants. We used bandwidth-based, height-based and curvature-based weighting schemes. Also, we tried to combine confidence measures by summation, by multiplication and by using only the testing vector's confidence values. These experiments were done by using both 25-NN and 50-NN classifiers. The results of this set of experiments are shown in figure 5.3:

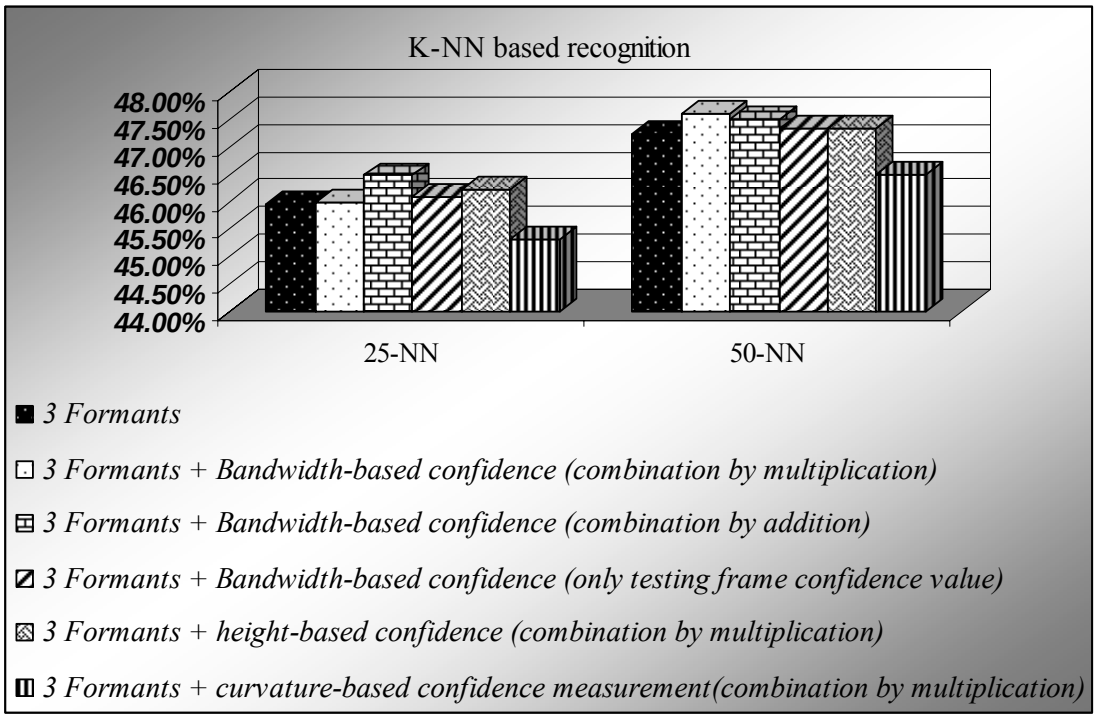

Figure 5. 3 The first 3 formants-based recognition performance by using confidence measures

The results above show that confidence-based recognition does not improve accuracy as expected. Confidence measures based on curvature extracted as described in [12] do not improve accuracy as stated. Confidence values that are normalized to sum up to N may introduce some noise. Maybe the normalization used in this experiment introduces noise in the distance measure. Also, combining confidence values by summation and multiplication does not seem to change results. Moreover, noise might be added by the combination methods. The best accuracy is achieved by using bandwidth-based confidence and by combining confidence measures in addition but this does not prove accuracy improvement because this improvement might be a statistical error.

The fourth set of K-NN experiments was done in order to improve recognition performance by using all formants. This recognizer type description is described in Chapter 4. Features distance is weighted by confidence measures based on the

sigmoid function described in Chapter 4. Also, we tried to combine sigmoid confidence measures with bandwidth-based confidence measures (combination by summation). Moreover, we used both Euclidean and Mahalanobis distance metrics.

In this set of experiments, we applied VTLN during the formants extraction process. We extracted all formants and tried to calculate all possible alignments between formants as described previously. Figure 5.4 shows the results of this set of experiments.

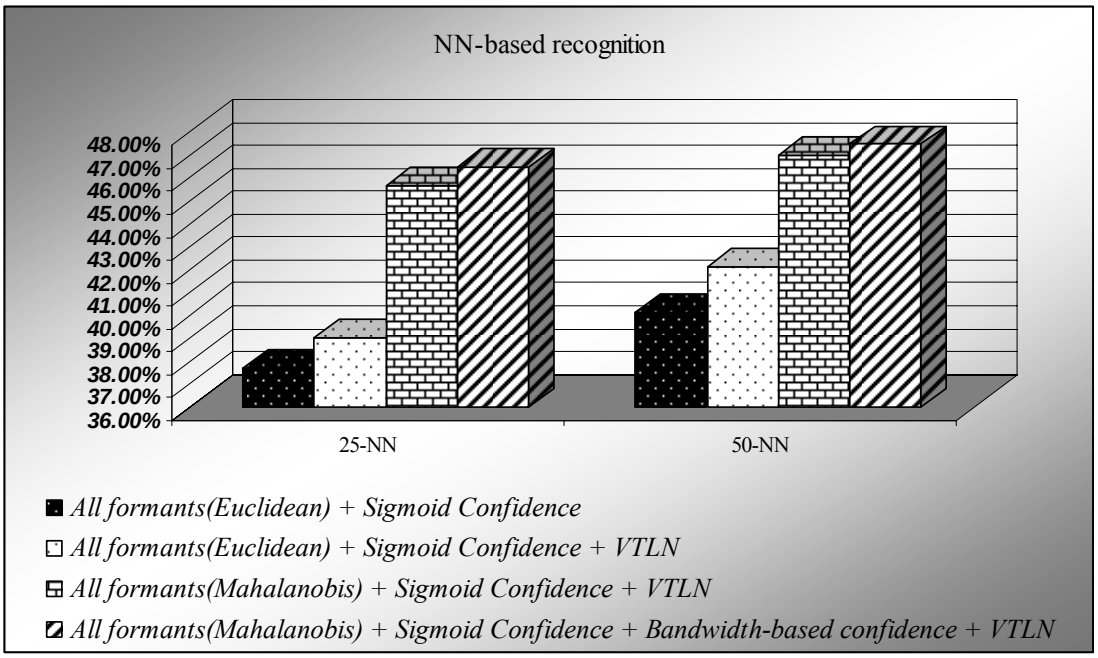

Figure 5. 4 formants recognition performance by using confidence measures and combing all possible formants alignments

Experiment results show that recognizers using mahalanobis distance metric outperform recognizers using Euclidean distance metric by ~6% higher accuracy. Also, VTLN application improves accuracy by ~2% as expected. Moreover, results show that bandwidth-based confidence improves performance slightly but the accuracy is always lower than 3-formant-based recognition. This shows that higher-order formants might not provide useful information to the recognition process.

5.2.4 Recognition performance by adding dynamic features

These set of experiments were the last K-NN based experiments. The purpose of these experiments was to examine the recognizer's behavior when adding some dynamic features, i.e. first and second derivatives, in the feature set. In order to extract derivatives, we used various combinations of Θ values (as described in chapter 4). Also, we applied VTLN during the features extraction process. The feature set that was used was consisted of three formants.

Furthermore, this procedure was repeated by using 12 MFCCs as features. Also, this procedure was done for both 25-NN and 50-NN classifiers. Figure 5.5 shows the results of this set of experiments.

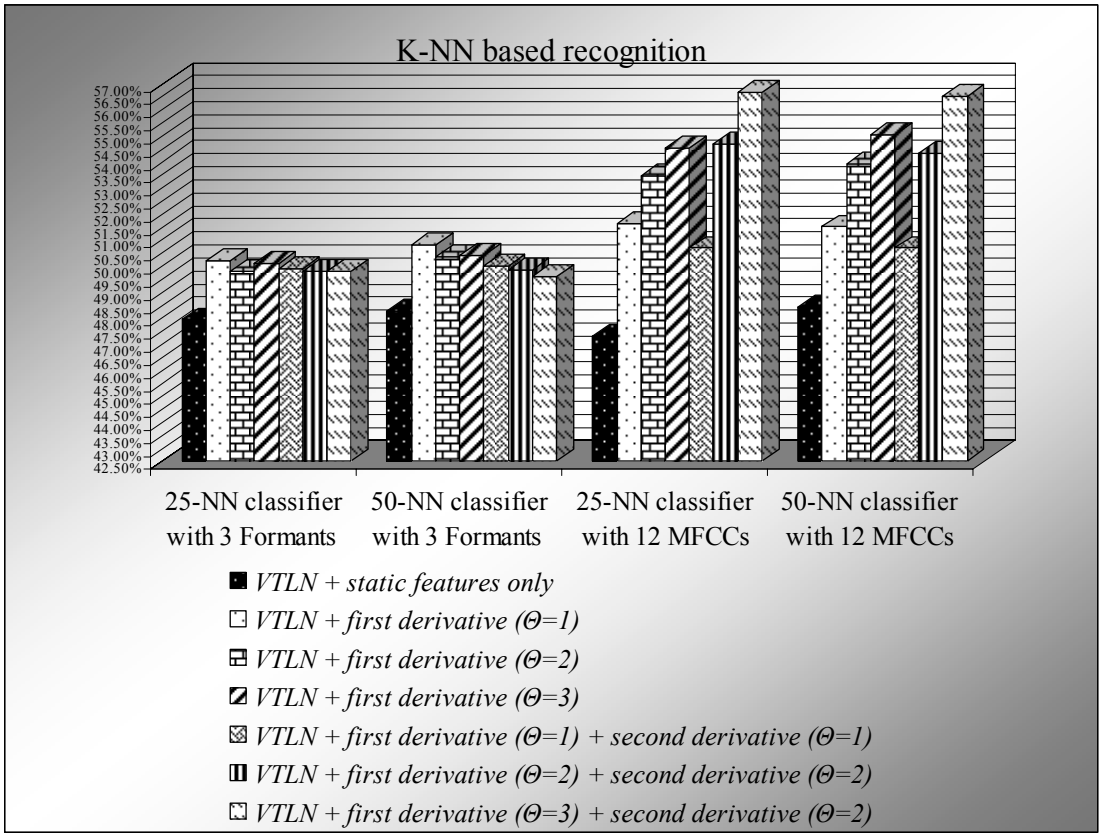

Figure 5. 5 The performance of K-NN-based classifiers that include dynamic features

This set of experiments shows that recognition performance is really improved with the addition of derivates in the feature set. The higher accuracy is achieved by using 12 MFCCs as static features with their first and second derivatives, along with Θ=3 and Θ=2 respectively. The reason of this improvement is that by including derivatives we incorporate important information about how the features change in time. Formants also seem to help, but they seem to be affected by noisy formants. It is obvious that by using more frames to extract derivates the possibility to find a frame with missing features is getting bigger and thus the probability to calculate a noisy derivative is higher. This is the reason that by using more frames in the derivative extraction process formants perform worst. MFCCs are more robust in the presence of noise and they perform better as we increase the number of frames involved in the calculation of derivatives.

5.3 HTK-based Experiments.

5.3.1 Recognition performance of various combinations of features

In the first set of HTK-based experiments, we did phoneme recognition by using various combinations of features in order to compare the accuracy achieved by using those features. The combinations of features used are 3 MFCCs, 3 formants, 12 MFCCs and 12 MFCCs along with 3 formants. These experiments were done by using one and three Gaussian mixtures as output probabilities of each emitting state. Figure 5.6 shows graphically the results of the first set of HTK-based experiments.

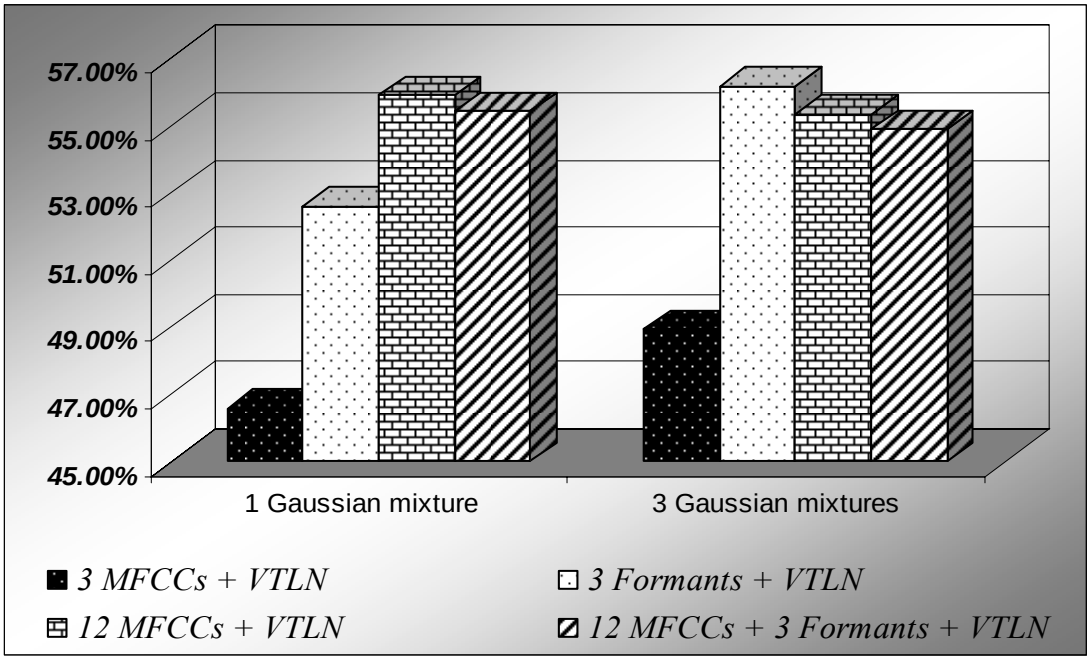

Figure 5. 6 The HTK-based recognition performance by using various combinations of features

It is clear that 3 formants-based recognizer performs better than 3 MFCC-based recognizer. Also, 3 formants-based recognizer in the 3 Gaussian mixtures case gives the best accuracy. Formants-based recognition performance is improving greatly by increasing the number of Gaussian mixtures. Possibly, this happens because recognition with more Gaussian mixtures reduces the influence of noise which is the main drawback of the formants. Another advantage of formants-based recognition is that it is faster than MFCCs-based recognition because we use considerably less features in the training and the testing procedures.

5.3.2 Recognition performance of vocal tract length normalization

This set of experiments was done to determine the effect of VTLN application in the accuracy of the recognized phonemes. These experiments were done by using three Formants and twelve MFCCs as features. Also, we tried the same experiment with output probabilities consisted of one and three Gaussian mixtures.

In order to estimate the warping factor in this experiment we used the pitch based-VTLN. The results of this experiment are shown in figure 5.7

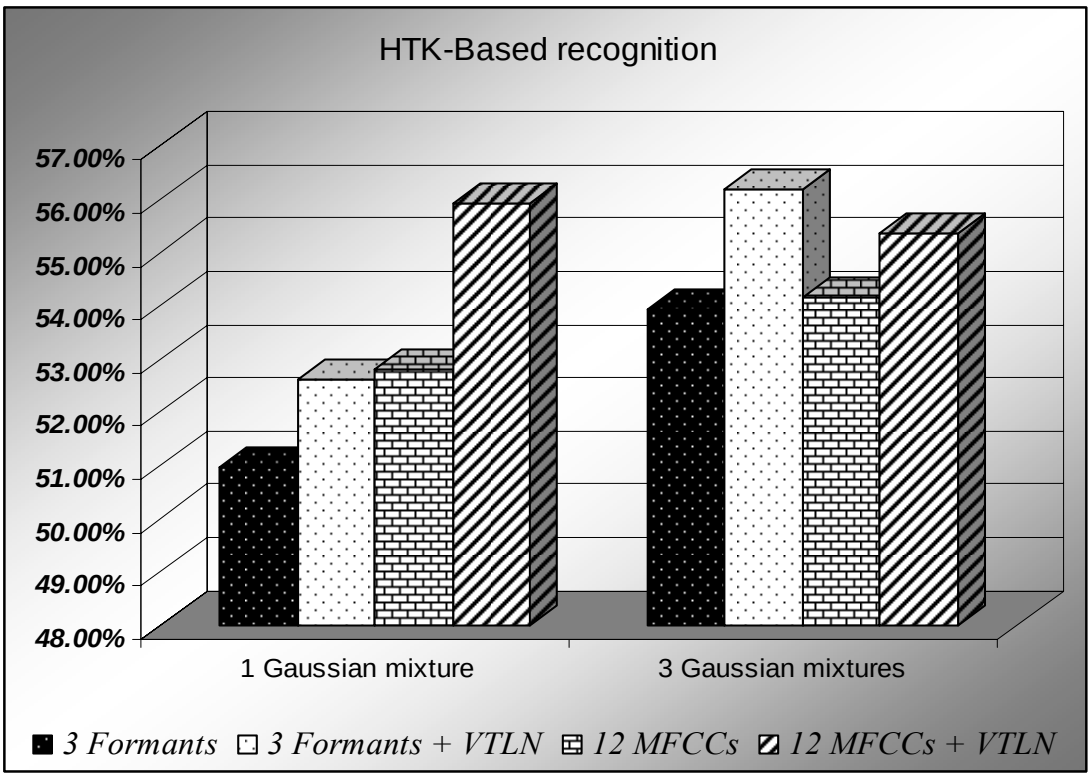

Figure 5. 7 The recognition performance by applying VTLN

By examining the above results, it is clear that the application of the pitch-based VTLN improves the performance of the recognizer and gives a maximum improvement of ~3%. Also, VTLN seems to improve results more in the case of one Gaussian mixture than in the case of three Gaussian mixtures. This might happen because, in the training process, the three Gaussian mixtures try to adapt to a multi-speaker environment and this might reduce the effect of VTLN.

5.3.3 Performance of formants-based recognition by using confidence measures

Another set of HTK-based experiments were done in order to improve accuracy rates of formant-based recognition. As said in theory, formants are sensitive to noise, so in this experiment we tried to integrate confidence measures in the distance calculation. In this set of experiments we used confidence measures based on bandwidth, height, curvature and bandwidth cut-off frequency. No combination method is needed because in these experiments we use only the confidence values of the testing feature vectors.

In all experiments of this set, we applied vocal tract normalization during the extraction process. As said above, the features that we used were the first three

formants. These experiments were done by using 1 Gaussian mixture as output probability. Figure 5.8 shows the results of this set of experiments.

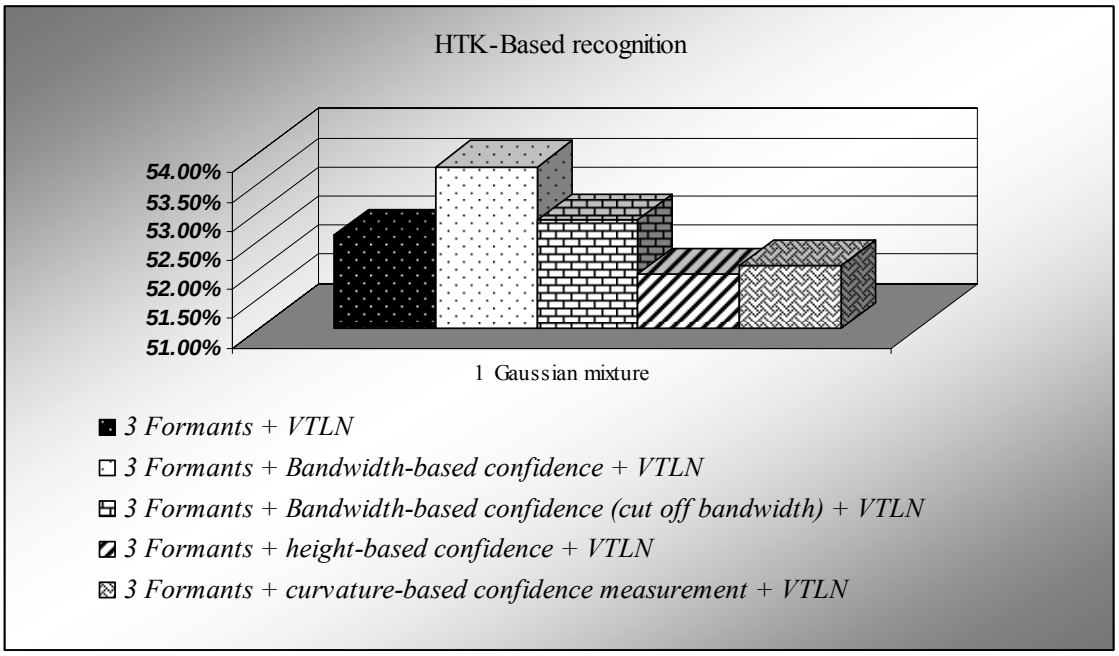

Figure 5. 8 The performance of formants-based recognition by using confidence measures

Generally, the above confidence types do not seem to help recognition accuracy significantly. Although, bandwidth-based confidence seems to increase slightly (~1%) recognition accuracy, it could be considered a statistical error. It is denotable that curvature based confidence does not improve accuracy as stated in [12]. It seems that before weighting each formant we have to find a way to correlate formants with its real label (first formant, second formant etc…).

5.3.4 Recognition performance by adding dynamic features

The last set of HTK-based experiments that we did aimed to examine the recognition performance of phonemes when we include dynamic features such as the first derivative. In order to do this set of experiments, we used as static features the first three formants. We calculated derivatives for each feature by using various Θ-values in order to check the behavior of the recognizer in each case.

Moreover, we repeated the above experiments by using 12 MFCCs as static features and by using output probabilities with 3 and 6 Gaussian mixtures. Figure 5.9 represents graphically the results of the above experiments.

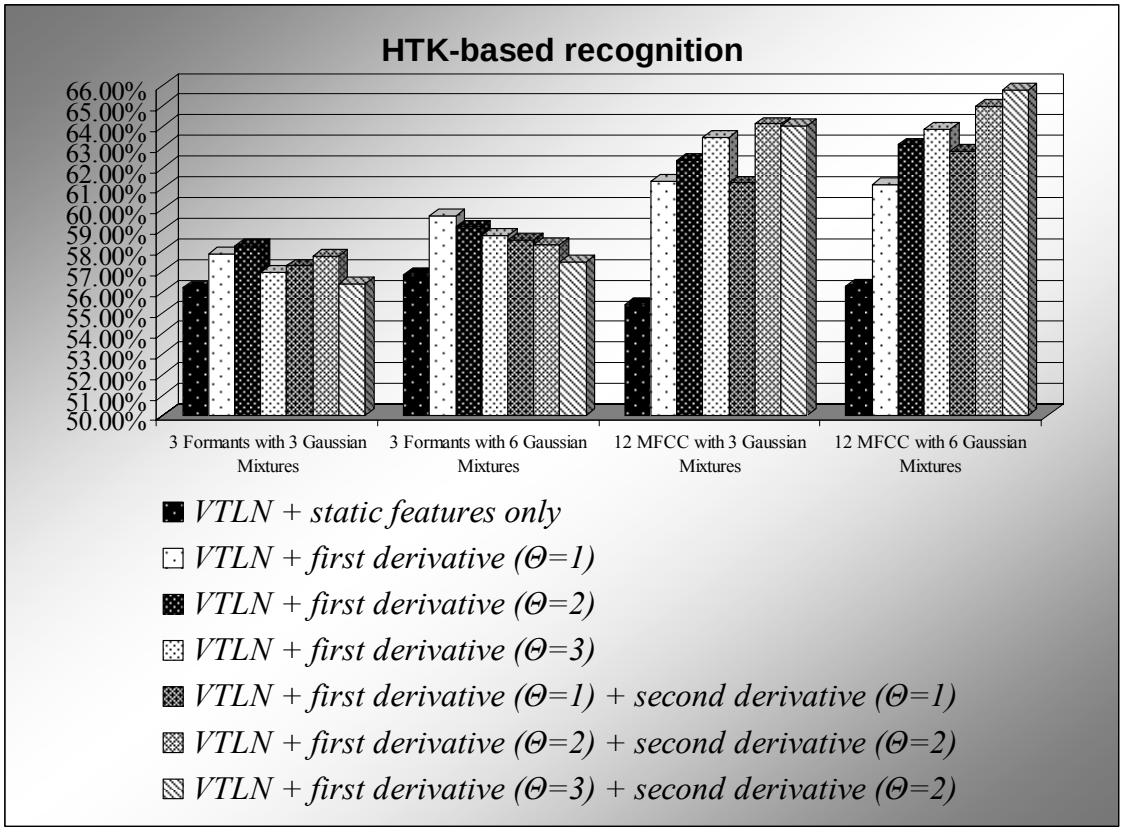

Figure 5. 9 The recognition performance by adding dynamic features

Generally, by including dynamic features in the feature set, recognition accuracy is improved. It is clear that the higher accuracy is achieved by using 12 MFCCs as static features and by using first and second derivatives with Θ=3 and Θ=2 respectively. The reason that MFCCs-based recognition performs better than Formant-based recognition is that Formants are affected more by the presence of noise than MFCCs. By increasing the number of frames that is included in the derivative calculation it is more probable to include a frame that misses, for example, the first formant. Thus, it is more probable that formants derivatives will be corrupted, leading to lowest accuracy rates.

5.4 SUMMARY

In this section, we introduced accuracy measures that we used to evaluate various types of speech recognizers. Moreover, we presented the results of many experiments for frames and phonemes recognition. We did various experiments by using different features such as MFCCs and formants. Furthermore, we did some experiments to investigation the performance of the VTLN. Another set of experiments was done in order to improve the performance of the formant-based recognizers by integrating confidence measures in the distance calculation process. In the last set of experiments that we did, we added dynamic features in the feature's vector and we were leaded to some valuable conclusions.

6 CONCLUSIONS and FUTURE WORK

6.1 Conclusions

The completion of this work, leads to some useful conclusions. Firstly, we saw that formants-based recognizers perform as good as MFCCs-based recognizers. However, formant-based recognizers are faster because they use smaller features vectors. On the other hand formant-based recognizers are easily affected by the presence of noise. This is obvious in phonemes such as 'ix' where MFCCs-based recognizers perform much better than formant based recognizers and the main reason is that, due to noise, formants are missing, leading to disastrous results. Also, we saw that in HTK experiments with multi-mixture output probabilities, the increase of the performance of the formants-based recognizer is higher than the MFCCs-based recognizer. This happens because multi-mixture cases can be trained to include noise effects, i.e. those provided from missing features.

Furthermore, we examined the effect of vocal tract length normalization. in various recognizers. We saw that VTLN increases recognition accuracy by 2-3%. This stands for both frame recognition and phoneme recognition experiments. Also, it is denotable that as we increase the number of Gaussian mixtures, the VTLN improvement on MFCCs-based recognition is lower than the 1 Gaussian mixture increase. A possible reason that this happens is that the multi-mixture model is already adapted to multi-speaker environments.

In addition, we experimented with various methods to deal with the problems of formants-based recognition. We tried to ignore missing features with soft-decoding. In order to do this, we tried to associate a confidence measure with each feature and then we use this measure to weight the distance between the testing and the training feature. In order to calculate confidence values, we used various methods, including bandwidth-based, height-based and curvature based methods. Moreover, we tried various methods to combine confidence measures of the testing and the training set, including combination by summation and multiplication. We concluded that this method does not improve the recognition accuracy significantly and presented various reasons that might cause this.

Additionally, we tried to improve the recognition accuracy by using all formants as features. In order to do this, we tried to find the distances of all possible alignments and we used sigmoid confidence measures. This set of experiments was done only by using the K-NN-based recognizer because we cannot train HMM models by using varying length feature vectors.

Lastly, we experimented with dynamic features, i.e. first and second derivatives. We concluded that addition of dynamic features improved MFCCs-based recognition. MFCCs-based recognition is not affected by noise as much as formants-based does. Formant-based recognition was also improved when we added first derivatives. By increasing the number of frames involved in the derivatives estimation, the noise produced by the corrupted formants increased too and results became worst.

During this work, the contribution of HTK toolkit was invaluable due to its speed and to its options. This toolkit is open-source and this gave us the chance to change the code and integrate weighting schemes during the calculation of the output probability.

6.2 Future work

Our work is on a relatively new research area. Formants-based recognition seems promising and there is much research work that has to be done in order to improve their recognition performance. Much work has to be done to be able to distinguish which formants are missing or added by noise.

6.2.1 Formants labeling.

Formants that are labeled incorrectly produce disastrous results in the recognition process and thus we have to find a way to label them correctly. In this work we tried to do this by assigning confidence measures to each feature. We can try other ways to connect confidence measures with the bandwidth and height. A simple way that might help is to try to find an empirical distribution that best fits to the training optimal weights. Another way is to try to estimate parametric distribution by Maximum Likelihood techniques.

6.2.2 Formants alignment

Another important problem that we have to deal with is the problem of calculating the distance of different formants. Work has to be done in order to solve this problem. An idea is to find a parametric distribution that connects the frequency of the formants with the most possible alignment and thus we can do this preprocessing step before starting the recognition process.

6.2.3 Confidence combination

Moreover, the methods for confidence combination presented here do not seem to work rather well. We have to search various methods of combining confidence values, such as taking the minimum or the maximum value or even estimating a parametric function that can predict the optimal confidence value that best describes the combination of the testing and the training confidence measure.

6.2.4 Weighting of formants during the derivative calculation

Experiments involving dynamic features showed that we have to weight formants during the slope estimation. This is a good research problem and a first approach

could be to use the weighted least square method and weight the formants during the slope estimation by a function inversely proportional to each formant variance.

6.2.5 Confidence values for dynamic features

It is also interesting to examine ways to extract confidence values for derivatives. This could significantly improve recognition results because derivatives are greatly affected from noisy formants and by weighting "clear" derivatives we can include only dynamic features that can improve the performance.

6.2.6 Vocal Tract Normalization based on Maximum Likelihood

In almost all experiments we used pitch based VTLN. We can train complete models in order to estimate warping factors by using Maximum Likelihood VTLN as presented in [7] and redo all the experiments to examine the results by using the optimal warping factors.

Formants-based recognition is a new research area that seems promising and can help to improve the performance of speech recognizers. There are much to be done before using them in speech recognition systems because in a real environment, that would probably be noisier than our testing phonemes environment, results could be worst.

APPENDIX A

A.1 Introduction

A.1.1 What is HTK?

HTK is the "Hidden Markov Model Toolkit" developed by the Cambridge University Engineering Department (CUED). This toolkit aims at building and manipulating Hidden
Markov Models (HMMs). HTK is primarily used for speech recognition research HTK consists of a set of library modules and tools available in C source form.

A.1.1 Recognition System

In this tutorial, we will build a 2-word recognizer with a {"Yes", "No"} vocabulary and silence {"sil"}

A.2 Creation of the Training Corpus

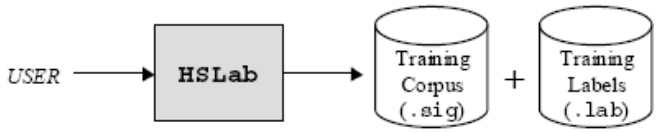

Fig A.1 Recording and labeling training data([26])

In order to create and label a speech file, we use the command:

HSLab any_name.sig

and the tool's graphical interface appears.

In order to record the signal, press "Rec" button to start recording the signal, then "Stop".
To label the speech waveform, first press "Mark", then select the region you want to label. When the region is marked, press "Labelas", type the name of the label, then press Enter. When the labels have been written, press "Save".

```
4171250 9229375 sil
9229375 15043750 yes
15043750 20430625 sil
```

Figure A.2 .lab file created that we created, Numbers indicate the start and end sample of each label([26])

A.3 Acoustical Analysis

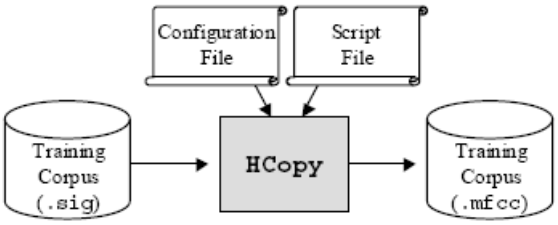

Fig. A.3: Conversion of the training data. ([26])

In this step, the signal is segmented in successive frames (whose length is chosen between 20ms and 40ms, typically), overlapping with each other. Then, each frame is multiplied by a Hamming function and a vector of acoustical coefficients (giving a compact representation of the spectral properties of the frame) is extracted from each windowed frame.

The conversion from the original waveform to a series of acoustical vectors is done with
the HCopy HTK tool:

HCopy -A -D -C analysis.conf -S targetlist.txt

analysis.conf is a configuration file
targetlist.txt specifies the name and location of each waveform to process, along with the name and location of the target coefficient files.

A.3.1 Configuration file

The configuration file is a text file. The configuration file used in this tutorial is the following:

```
SOURCEFORMAT = HTK              # Gives the format of the speech files
TARGETKIND = MFCC_0_D_A         # Identifier of the coefficients to use

# Unit = 0.1 micro-second :
WINDOWSIZE = 250000.0           # = 25 ms = length of a time frame
TARGETRATE = 100000.0           # = 10 ms = frame periodicity

NUMCEPS = 12                    # Number of MFCC coeffs (here from c1 to c12)
USEHAMMING = T                  # Use of Hamming function for windowing frames
PREEMCOEF = 0.97                # Pre-emphasis coefficient
NUMCHANS = 26                   # Number of filterbank channels
CEPLIFTER = 22                  # Length of cepstral liftering
```

Figure A.4 The configuration file is a text file([26])

With such a configuration file, an MFCC analysis is performed. For each signal frame, the coefficients that are extracted are the 12 first MFCC coefficients, the MFCC coefficient c0, 13 “Delta coefficients”, 13 “Acceleration coefficients”.

A.3.2 Source / Target Specification

the –S option is used to specify One or more "source file / target file" pairs.

```
data/train/sig/yes01.sig   data/train/mfcc/yes01.mfcc
data/train/sig/yes02.sig   data/train/mfcc/yes02.mfcc
      etc...
data/train/sig/no01.sig   data/train/mfcc/no01.mfcc
data/train/sig/no02.sig   data/train/mfcc/no02.mfcc
```

A.5 Conversion script file. ([26])

A.4 HMM Definition

In this tutorial, 3 acoustical events have to be modeled with a Hidden Markov Model (HMM): "Yes", "No" and "Silence". For each one we will design a HMM. The first step is to choose a priori for each HMM the number of states, the form of the observation function and the disposition of transitions between states

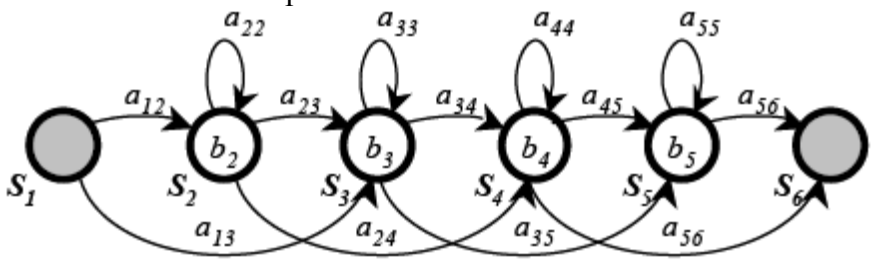

Figure A.7 Hidden Markov Model([26])

```
~o <VecSize> 39 <MFCC_0_D_A>
~h "yes"
<BeginHMM>
      <NumStates> 6
      <State> 2
         <Mean> 39
           0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
           0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
           0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
         <Variance> 39
           1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
           1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
           1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
      <State> 3
         <Mean> 39
           0.0 0.0 (...) 0.0
         <Variance> 39
           1.0 1.0 (...) 1.0
      <State> 4
         <Mean> 39
           0.0 0.0 (...) 0.0
         <Variance> 39
           1.0 1.0 (...) 1.0
      <State> 5
         <Mean> 39
           0.0 0.0 (...) 0.0
         <Variance> 39
           1.0 1.0 (...) 1.0
      <TransP> 6
           0.0 0.5 0.5 0.0 0.0 0.0
           0.0 0.4 0.3 0.3 0.0 0.0
           0.0 0.0 0.4 0.3 0.3 0.0
           0.0 0.0 0.0 0.4 0.3 0.3
           0.0 0.0 0.0 0.0 0.5 0.5
           0.0 0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Figure A.8 HMM prototype([26])

~o <VecSize> 39 <MFCC_0_D_A>
is the header of the file, giving the coefficient vector size , and the type of coefficient.

~h "yes" <BeginHMM> (...)<EndHMM>
encloses the description of a HMM called “yes”.

<NumStates> 6
gives the total number of states in the HMM, including the 2 non-emitting states 1 and 6.

<State> 2
introduces the description of the observation function of state 2. Here we have chosen to use single-gaussian observation functions, with diagonal matrices.

<Mean> 39
0.0 0.0 (...) 0.0 (x 39)
gives the mean vector (in a 39 dimension observation space) of the current observation function.
<Variance> 39
1.0 1.0 (...) 1.0 (x 39)
gives the variance vector of the current observation function.
<TransP> 6
gives the 6x6 transition matrix of the HMM,

Such a prototype has to be generated for each event to model.

A.5 HMM Training

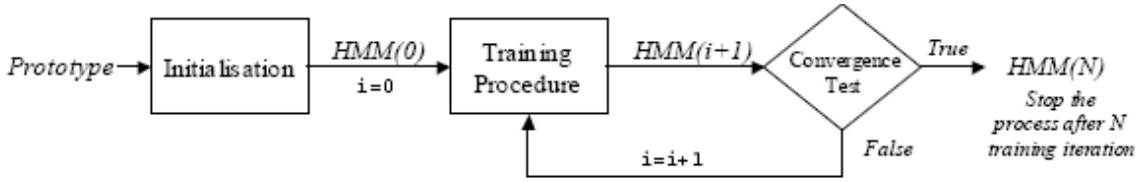

Figure A.9: Complete training procedure([26])

A.5.1 Initialization

Before starting the training process, the HMM parameters must be properly initialised with training data in order to allow a fast and precise convergence of the training algorithm.

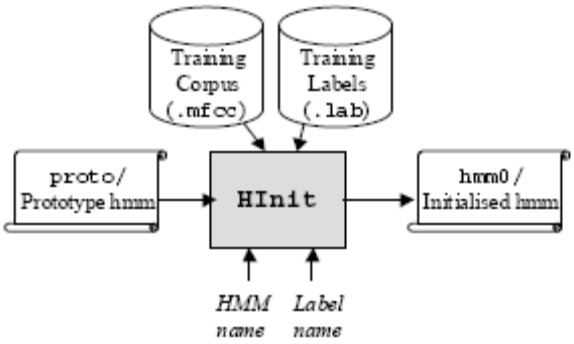

Figure A.10: Initialization from a prototype([26])

HInit
The following command line initialises the HMM by time-alignment of the training data
with a Viterbi algorithm:
HInit -A -D –T 1 -S trainlist.txt -M model/hmm0 \
-H model/proto/hmmfile -l label -L label_dir nameofhmm

- nameofhmm is the name of the HMM to initialise
- hmmfile is a description file containing the prototype of the HMM
- trainlist.txt gives the complete list of the .mfcc files forming the training corpus
- label_dir is the directory where the label files corresponding to the corpus
- label indicates which labelled segment must be used within the training
- model/hmm0 is the name of the directory where the resulting initialised HMM description will be output.

This procedure has to be repeated for each model (hmm_yes, hmm_no, hmm_sil).

A.5.2 Training

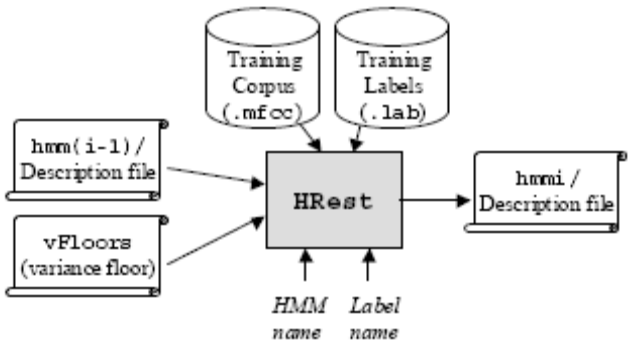

Figure A.11: A re-estimation iteration. ([26])

The following command line perform re-estimation iteration with HTK tool HRest, estimating the optimal values for the HMM parameters:

```
HRest -A -D -T 1 -S trainlist.txt -M model/hmmi \
-H model/ /hmmfile -l label -L label_dir nameofhmm
```

- nameofhmm is the name of the HMM to train
hmmfile is the description file of the HMM called nameofhmm.
trainlist.txt gives the complete list of the .mfcc files forming the training corpus
label_dir is the directory where the label files corresponding to the corpus
label indicates the label to use within the training data
model, the output directory

A.6 Grammar and Dictionary

In HTK, the task grammar is written in a text file, according to some syntactic rules.

```
$WORD = YES | NO;

( { START_SIL } [ $WORD ] { END_SIL } )
```

Figure A.12 Basic task grammar([26])

The WORD variable can be replaced by YES or NO. The brackets {} around START_SIL and END_SIL denotes zero or more repetitions The brackets [] around $WORD denotes zero or one occurrence.

The system must of course know to which HMM corresponds each of the grammar variables YES, NO, START_SIL and END_SIL. This information is stored in a text file called the task dictionary.

```
YES         [yes]   yes
NO          [no]    no
START_SIL   [sil]   sil
END_SIL     [sil]   sil
```

Figure A.13 Basic task dictionary([26])

The left elements refer to the names of the task grammar variables. The right elements refer to the names of the HMMs. The bracketed elements in the middle are optional, they indicate the symbols that will be output by the recognizer.

A.7 Network

The task grammar have to be compiled with tool HParse, to obtain the task network by using the command:

HParse -A -D -T 1 gram.txt net.slf

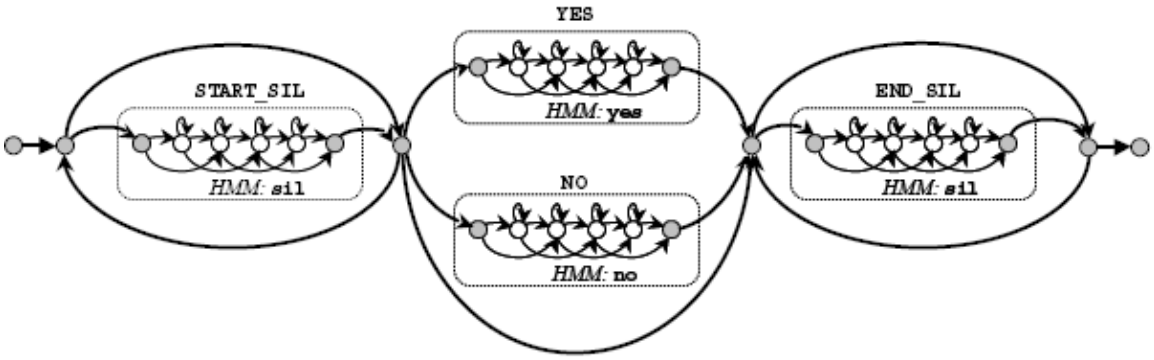

Figure A.14 Network + Dictionary + HMMs([26])

A.8 Recognition

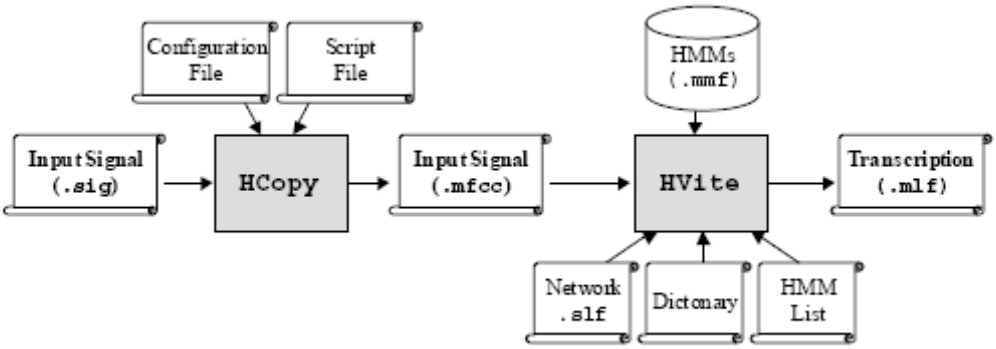

Figure A.15 Recognition of an unknown signal. ([26])

An input speech signal is transformed into a series of "acoustical vectors" with tool HCopy and the result is stored in an input.mfcc file. The input observation is then process by a Viterbi algorithm, which matches it against the recogniser's Markov models.

This is done by tool HVite:

```
HVite -A -D -T 1 -H hmmsdef.mmf -i reco.mlf -w net.slf \
dict.txt hmmlist.txt input.mfcc
```

- input.mfcc is the input data to be recognised.
- hmmlist.txt lists the names of the models to use.
- dict.txt is the task dictionary.
- net.slf is the task network.
- reco.mlf is the output recognition transcription file.
- hmmsdef.mmf contains the definition of the HMMs (called a Master Macro File, with extension .mmf). Such a file is simply obtained by copying each definition after the other in a single file.

```
~o <VecSize> 39 <MFCC_0_D_A>
~h "yes"
<BeginHMM>
              (definition...)
<EndHMM>
~h "no"
<BeginHMM>
              (definition...)
<EndHMM>
~h "sil"
<BeginHMM>
              (definition...)
<EndHMM>
```

Figure A.16 Master Macro File([26])

The output is stored in a file (reco.mlf) which contains the transcription of the input.

```
#!MLF!#
"../data/train/mfcc/yes01.rec"
0 4900000 SIL -2394.001465
4900000 12000000 YES -5159.434570
12000000 18300000 SIL -3289.197021
.
```

Figure A.17 Recognition Output([26])

A.9 Master Label Files

The master label file will contain the "correct" transcriptions of the whole test corpus, that is, the transcriptions obtained by hand-labelling. Let's call ref.mlf these reference transcriptions.

Each transcription is introduced by a file name and terminated by a period ".".

A Master Label File has the following structure:

```
#!MLF!#
"path/data01.ext"
Label1
Label2
.
"path/data02.ext"
Label1
Label2
Label3
Label4
.
```

Figure A.18 Master Label File([26])

A.10 Error Rates

The master transcriptions are compared with the HTK performance evaluation tool, HResults:

HResults -A -D -T 1sil -I ref.mlf labellist.txt results.txt

- results.txt contains the output performance statistics.
- labellist.txt lists the labels appearing in the transcription files.
- ref.mlf contains the reference transcriptions of the test data.

```
====================== HTK Results Analysis =======================
  Date: Tue Dec 03 19:12:58 2002
  Ref : .\ref.mlf
  Rec : .\rec.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=80.00 [H=8, S=2, N=10]
WORD: %Corr=80.00, Acc=80.00 [H=8, D=0, S=2, I=0, N=10]
====================================================================
```

Figure A.19 Master Label File([26])

The figure above shows an example of the kind of results that can be obtained. The first line gives the sentence recognition rate (%Correct=80.00), the second one (WORD) gives the word recognition rate (%Corr=80.00).

APPENDIX B

B.1 Recognition Results of an HTK-based experiment

The feature set includes the first three formants extracted by using VTLN and output probability consists of one Gaussian Mixture

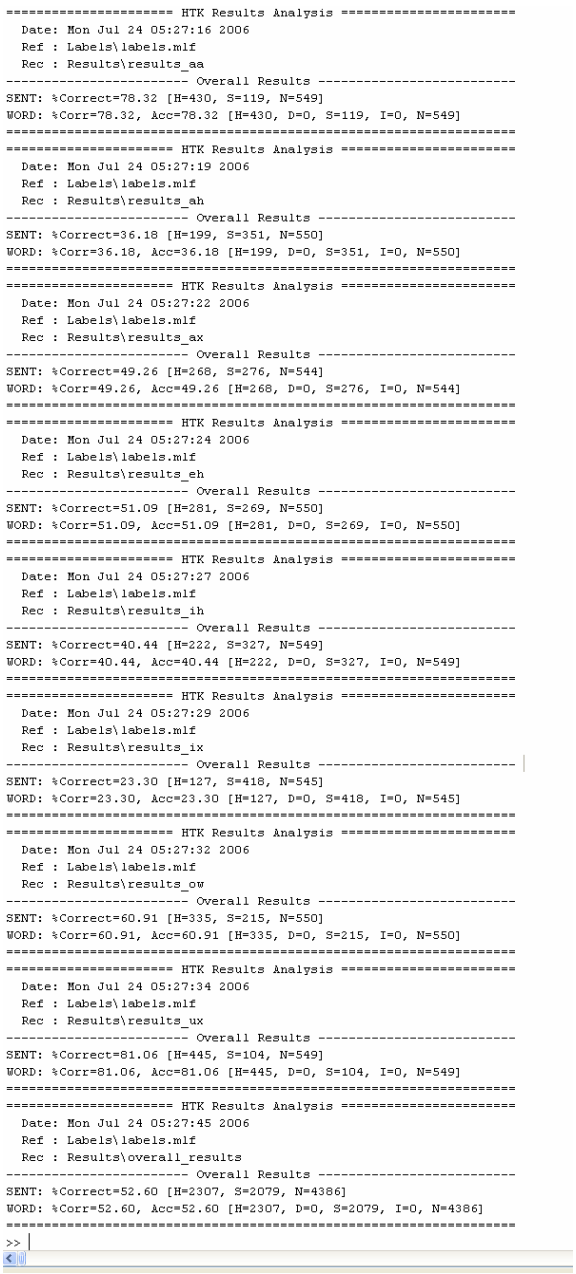

Figure B.1 Results from an HMM-based experiment

B.2 Recognition Results of an K-NN-based experiment

The feature set includes 12 MFCCs extracted by using VTLN.

	NN-50	NN-25
phoneme aa	70.30%	68.19%
phoneme ah	22.05%	22.78%
phoneme ax	14.77%	15.19%
phoneme eh	54.28%	51.99%
phoneme ih	31.86%	32.84%
phoneme ix	7.82%	8.98%
phoneme ow	56.36%	54.09%
phoneme ux	79.08%	76.60%
overall	48.39%	47.30%

Figure B.2 Results from a K-NN-based experiment

APPENDIX C

C.1 More K-NN-based figures of Evaluation

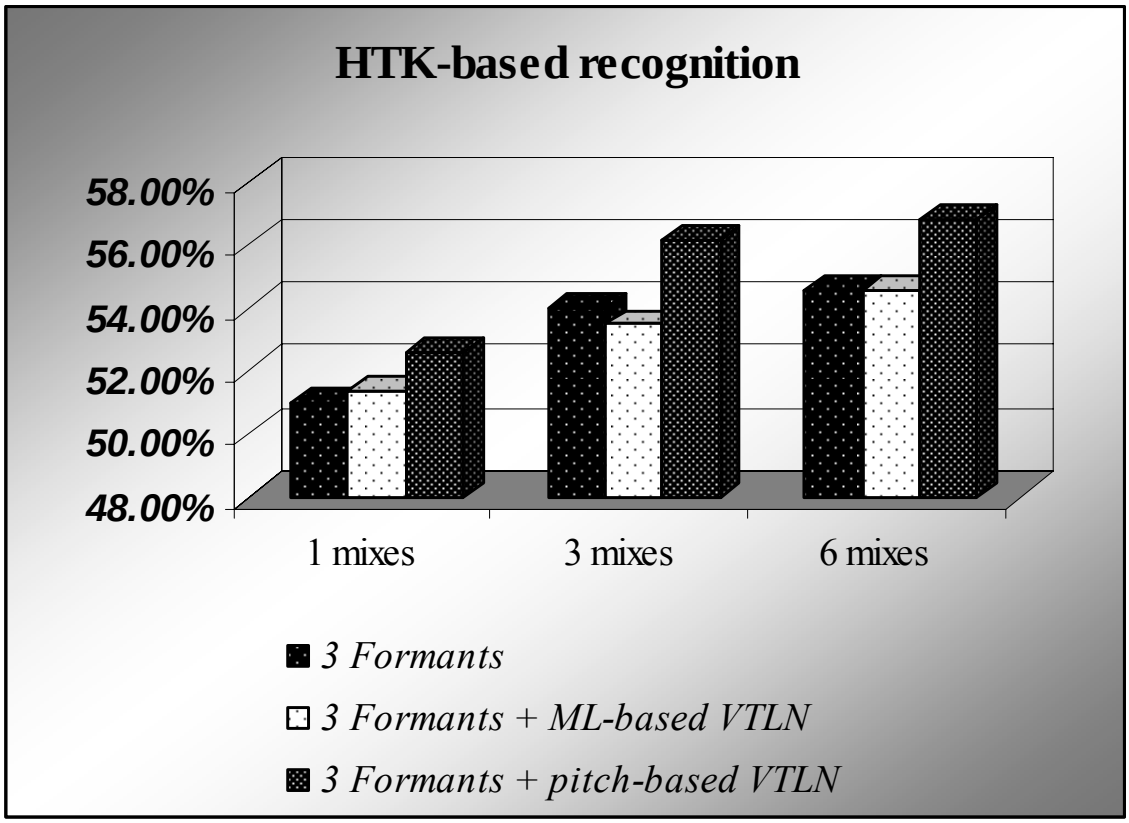

Figure C.1 Comparison of VTLN techniques

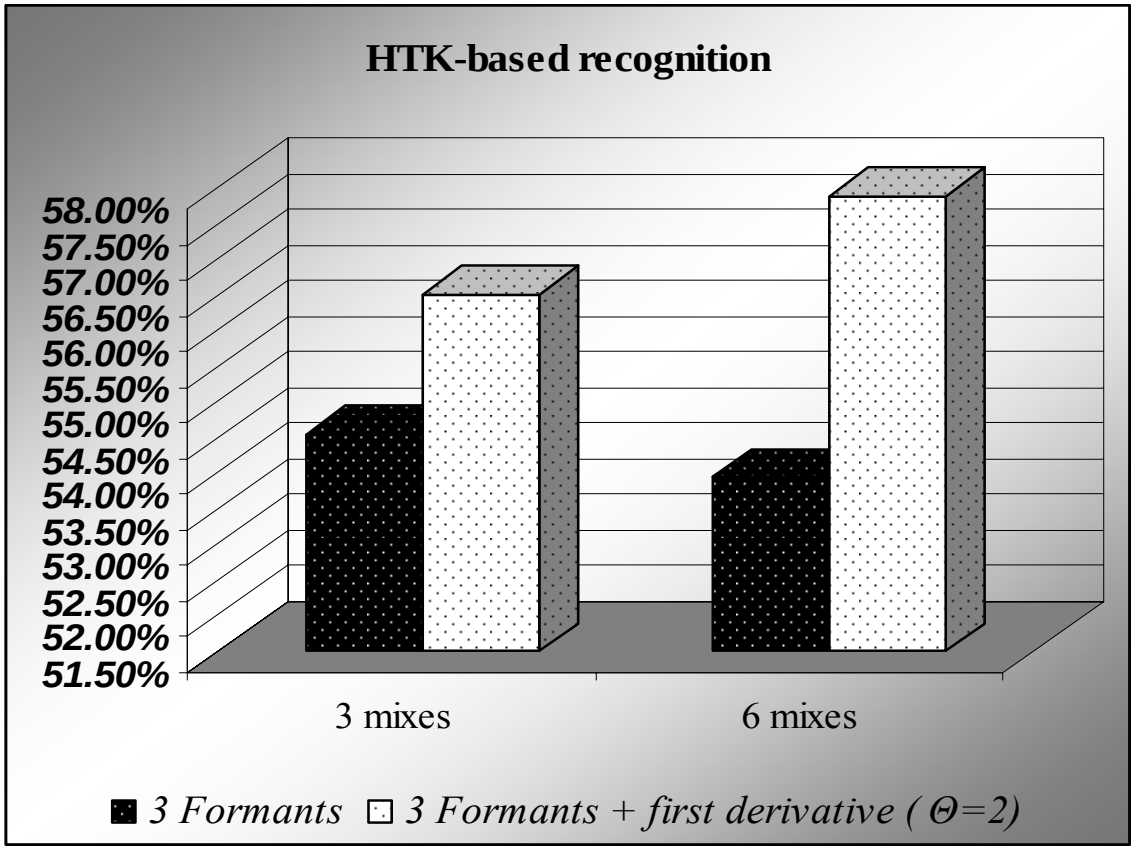

Figure C.2 Addition of first derivatives without VTLN

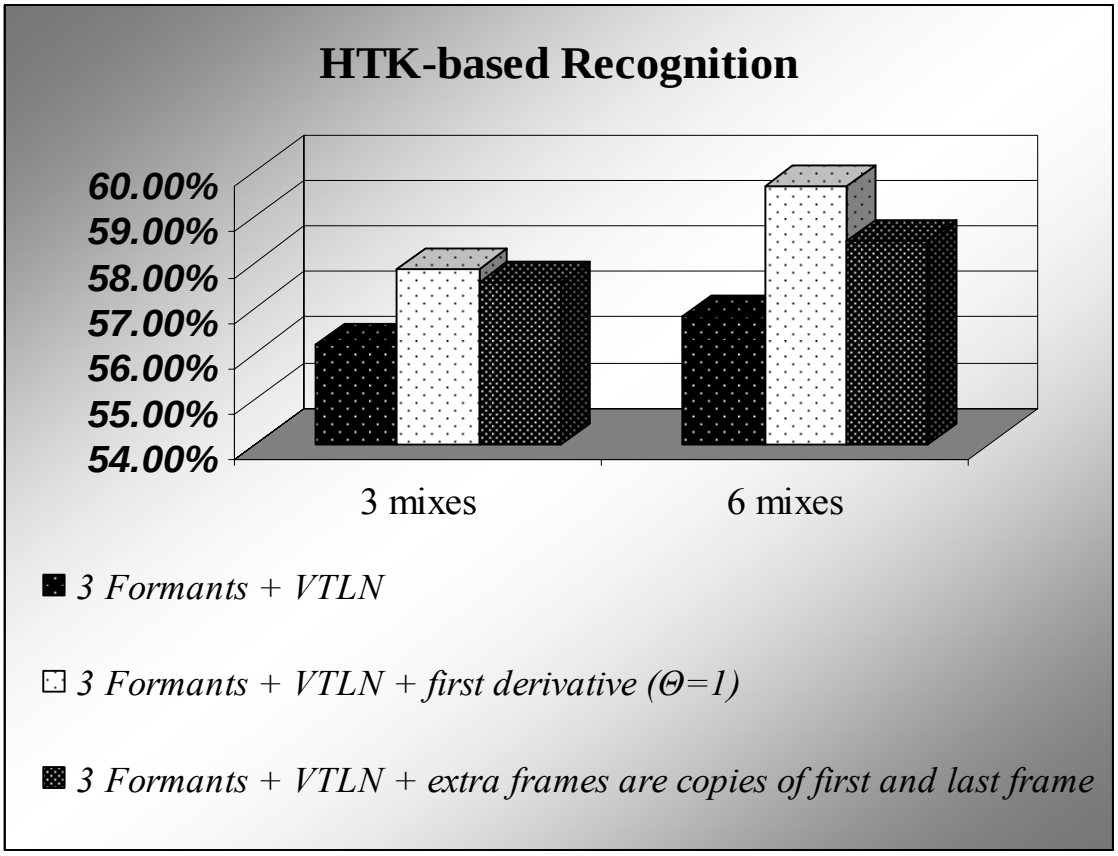

Figure C.3 Comparison of derivatives extraction techniques

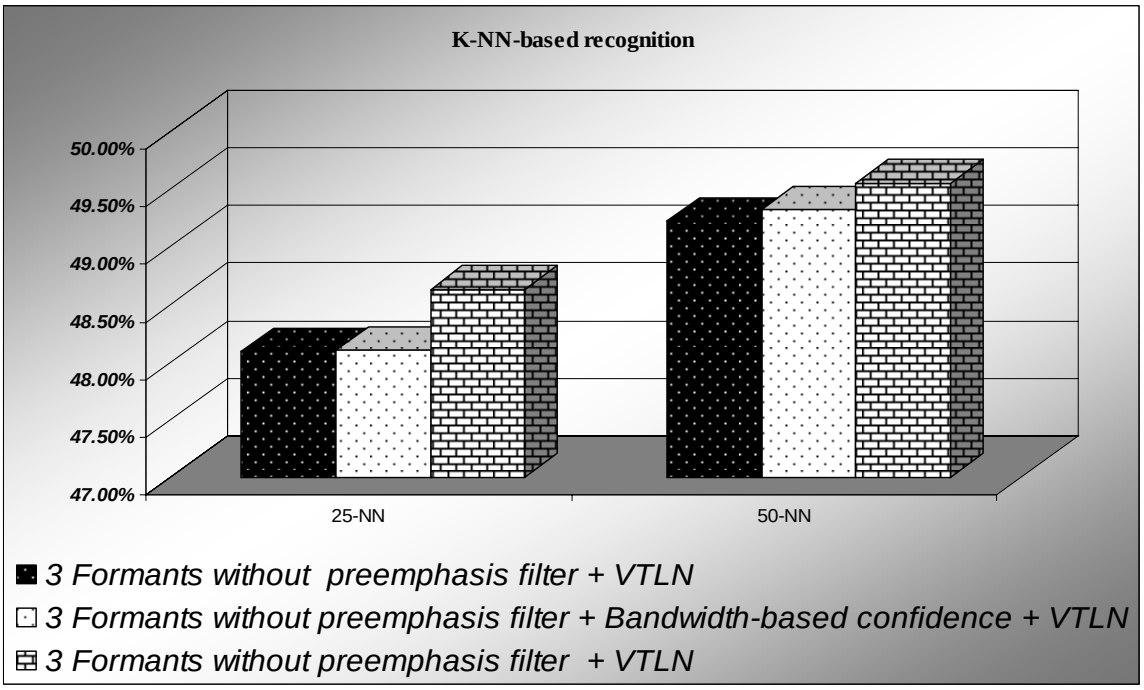

Figure C.4 Recognition without applying preemphasis filtering

BIBLIOGRAPHY

[1]Claudio Becchetti, and Lucio Pina Ricotti., “Speech Recognition. Theory and C++ Implementation.”, John Wiley & Sons, 1999.

[2]Lawrence Rabiner, and Bing-Hwang Juang, “Fundamentals of speech recognition.”, Prentice Hall, 1993.

[3]L.R. Rabiner, and R.W. Schafer “Digital Processing of Speech Signals.” Prentice Hall, Signal Processing Series, 1978.

[4]Βασίλης Διγαλάκης, “Σημειώσεις εισαγωγής του μαθήματος Εισαγωγή στην επεξεργασία φωνής.”, Πολυτεχνείο Κρήτης.

[5]R.Duda, D. Stork, and P. Hart “Pattern Classification”, John Wiley & Sons, 2000.

[6]Steve Yound, Gunnar Evermann, Mark Gales, Thomas Hain, Den Kershaw, Gareth Moore, Juliann Odell, Dave Ollason, Dan Povey, Valtchio Valtchev, and Phil Woodland, “The HTK Book.”, Cambridge university Engineering department, 2005.

[7]Li Lee, and Richard Rose, “A Frequency Warping Approach to Speaker Normalization.”, IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING, Vol6,JANUARY ,1998.

[8]Arlo Faria, “Pitch-based Vocal Tract Length Normalization.”, Tech. Rep. TR-03-001, International Computer Science Institute, NOVEMBER 2003.

[9]N.J. Wilkinson, and M.J. Russel “Improved Phone Recognition on TIMIT using Formant Frequency Data and Confidence Measures”, Proc. ICSLP, 2002.

[10]J.N Holmes, W.J Holmes and P.N. Garner, “Using formant frequencies in speech recognition”, Eurospeech 97 Rhodes, Greece, volume 4, pages 2083-2086, September, 1997.

[11]W.J. Holmes, and P.N. Garner, “On robust incorporation of formant frequencies into Hidden Markov Modelsfor automatic speech recogntion”, IEEE ICASP’ 98, Instanbul, Turkey, pages 1-4, May 1998.

[12]J.N. Holmes, “Speech processing system using formant analisis”, US patent US6292775, September 2001.

[13]Don Hnja, and Bruce R. Musicus, “The Solafs Time-Scale Modification Algorithm”, BBN Technical Report, July, 1991.

[14]“Matlab Help”, R14, 2004.

[15]P Zhan, and M Westphal, “Speaker Normalization based on Frequency warping.”,. ICASSP-97, 1997.

[16]Gouvêa, E.B., and Stern, R.M., “Speaker Normalization Through Formant-Based Warping of the Frequency Scale”, Proc. Eurospeech, Rhodes, 1997.

[17]K. Y. Leung and M. Siu, “Articulatory-feature-based Confidence Measure.” Computer, Speech and Languag, August, 2005

[18]Shah, J., Yantorno, R.E., Smolenski, B., Iyer, A.N., “Sequential K-Nearest Neighbor Pattern Recognition for Usable Speech Classification”, European Signal Processing Conference, EURSIPCO, 2004.

[19]Feldman, A., and Balch, T., “Modeling Honey Bee Behavior for Recognition Using Human Trainable Models”, Workshop at AAMAS, New York City, 2004

[20]Fabrice Lefevre, Claude Montacie, and Marie-José Caraty, “On the Influence of the Delta Coefficients in a HMM-based Speech Recognition System”, ICSLP’98, December, 1998.

[21]Claude Montacie, Marie-Jose Caray, and Fabrice Lefevre, “K-NN versus Gaussian in HMM-based recognition system.”, EUROSPEECH '97, Rhodes, Greece, September, 1997.

[22]M. Wolfel, H.K. Ekenel, "Feature Weighted Mahalanobis Distance: Improved Robustness for Gaussian Classifiers", EUSIPCO 2005, Antalya, Turkey, September 2005.

[23]K. Mustafa, “Robust Formant Tracking for Continuous Speech With Speaker Variability,” M.S. thesis, McMaster Univ., Hamilton, ON, Canada, 2003.

[24]De Wet, F., Weber, K, Boves, L., Cranen, B., Bengio, S., and Bourlard, “Evaluation of formant-like features for automatic speech recognition,” in Journal of Acoustical Society of America , Vol. 116, Isuue 3, pp. 1781-1792, September 2004

[25] Q. Yan, S. Vaseghi, E. Zavarehei, B. Milner, “FORMANT-TRACKING LINEAR PREDICTION MODELS FOR SPEECH PROCESSING IN NOISY ENVIROMENTS”, Interspeech 2005, Lisbon, 2005.

[26]Nicolas Moreau, “Basic HTK tutorial.”, 2002

[27]“Wikipedia”