République Algérienne Démocratique et populaire
Université M'hamed Bougara, Boumerdes
Faculté des Sciences de l'Ingénieur
Département de Génie Electrique et Electronique

Continuous Speech Recognition

Thesis
presented in partial fulfilment of the requirements of the degree of
Docteur es Sciences
in
Electronic System Engineering
by
Abdelhakim DAHIMENE

Before the jury composed of:

Dr. Aksas R. (Professeur, ENP)	President
Dr. Guessoum A. (Professeur, Université S. Dahlab, Blida)	Examiner
Dr. Trabelsi M. (Maître de conférences, ENP)	Examiner
Dr. Hariche K. (Professeur, UMBB)	Examiner
Dr. Noureddine (Maître de conférences, UMBB)	Supervisor

April 2009

Acknowledgments

I wish to express my sincere appreciation to Dr. Mohamed Noureddine for his guidance and encouragement throughout this thesis.

To my wife, Farida, I thank her for her infinite patience and love. I must not forget Nawel and Ryadh.

ملخص

يعد تمييز الكلام أحد أهم تحديات بداية هذا القرن. كما سمحت التطورات المذهلة للأجهزة الالكترونية باستعمال طرق رياضية غاية في التعقيد محل مشاكل جد مستعصية. في هذه المذكرة, سنحول أن نبرهن أن الطرق التسلسلية المخترعة لفك الرموز "convolutionnels" ذات العوائق جد الطويلة يمكن أن تستعمل في تمييز الكلام.

أهم المساهمات في هذه المذكرة يمكن تلخيصها فيما يلي:

1. تطبيق خوارزمية البطارية (التسلسلية) لتمييز الكلام المتواصل.
2. تطوير و تحليل مقياس للمسافة (داخل شجرة) بالاعتماد على مقياس المسافة لماهالانوبيس. هذا المقياس استخدم لبرمجة خوارزمية البطارية في برنامج التمييز و كذا في خوارزمية فيتربي في برنامج التلقين.
3. تطوير خوارزمية جديدة بناءا على التوقع الخطي لاسترجاع إشارة الكلام المبتورة,
4. تطوير طريقة لإزالة الضوضاء في إشارة الكلام بناءا على مصفاة وينر ذات درجة منخفضة.
5. للحصول على مقياس مقبول للمسافة, قمنا بتحليل إحصائي لثلاث تمثيلات وسيطية للكلام و أثباتنا أن معاملات (MFCC) تتبع توزيع دالة غوس و تعطي أفضل بين الأقسام مقارنة بمعاملات (LPC) و (PARCOR),
6. في آخر فصل من المذكرة, قمنا بتطوير خوارزمية آلية للتجزئة والتي نستعملها قي تلقين برامج تمييز الكلام.

Résumé

La reconnaissance de la parole continue est un des plus grands défis dans ce début du siècle. Les avances impressionnantes dans l'équipement électronique permettent l'utilisation de méthodes mathématiques sophistiquées pour résoudre des problèmes extrêmement complexes. Dans cette thèse, nous allons montrer que les méthodes séquentielles inventées pour décoder les codes convolutionnels de grande longueur de contrainte peuvent être utilisées dans la reconnaissance de la parole.

Les contributions principales de cette thèse peuvent être récapitulées comme suit:

(1) L'applicabilité de l'algorithme de décodage à pile (séquentiel) à la reconnaissance de la parole continue.

(2) Le développement et l'analyse d'une métrique de chemin (dans un arbre) basé sur la distance de Mahalanobis. Cette métrique a été utilisée dans la mise en place de l'algorithme de pile dans le programme de reconnaissance et également dans l'algorithme de Viterbi dans le programme d'apprentissage.

(3) Le développement d'un nouvel algorithme basé sur la prédiction linéaire pour la restauration d'un signal de parole écrêté.

(4) Une méthode de suppression du bruit dans le signal de parole basée sur des filtres de Wiener à paramètres non stationnaires. Nous avons obtenu des résultats remarquables à l'aide d'un filtre d'ordre très réduit.

(5) Afin de trouver une bonne métrique de chemin pour notre algorithme à pile, nous avons effectué une analyse statistique de trois représentations paramétriques de la parole et nous avons prouvé que les paramètres de MFCC sont pratiquement Gaussien et fournissent la meilleure séparabilité entre classes par rapport aux coefficients de LPC et de PARCOR.

(6) Dans le dernier chapitre de la thèse, nous avons développé un algorithme automatique de segmentation que nous utilisons pour l'apprentissage du programme de reconnaissance de la parole.

Abstract

Continuous speech recognition is one of the greatest challenges in this beginning of the century. The impressive advances in hardware allow the use of sophisticated mathematical methods to solve complex problems. In this thesis, we show that methods invented for solving long constraint length convolutional codes can be used in speech recognition.

The main contributions of this thesis can be summarized as follows:

(1) The applicability of the stack decoding algorithm to continuous speech recognition.

(2) The development and the analysis of a path metric based on the Mahalanobis distance. This path metric has been used in the implementation of the stack algorithm in recognition program and also in the Viterbi algorithm in the training program.

(3) The development of a novel algorithm for clipped speech restoration based on linear prediction.

(4) Speech denoising method based on time varying Wiener filters. We obtained remarkable results using a very low order filter.

(5) In order to develop a good path metric for our stack algorithm, we have performed a statistical analysis of three parametric representations of speech and we have shown that the MFCC set is nearly Gaussian and provides the best separability between classes as compared with LPC and PARCOR coefficients.

(6) In the last chapter of the thesis, we have developed an automatic segmentation algorithm that we use for training the speech recognition program.

Table of Contents

List of Illustrations

List of Tables

Abbreviations

ADC	Analog to Digital Converter
AGC	Automatic Gain Control
AR	AutoRegressive
ARMA	AutoRegressive Moving Average
ASR	Automatic Speech Recognition
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DRT	Diagnosis Rhyme Test
DTW	Discrete Time Warping
EBNF	Extended Backus-Naur Form
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FSA	Finite State Automaton
FSR	Finite State Recognizer
HMM	Hidden Markov Model
IDFT	Inverse Discrete Fourier Transform
IIR	Infinite Impulse Response
LPC	Linear Prediction Coding
MAP	Maximum A posteriori Probability
MFCC	Mel Frequency Cepstral Coefficients
ML	Maximum Likelihood
MMSE	Minimum Mean Square Estimation
PARCOR	Partial Correlation
PC	Personal Computer
PDF	Probability Density Function

PSD	Power Spectrum, Power Spectral Density
SFSA	Stochastic Finite State Automaton
SNR	Signal to Noise Ratio
STFT	Short Time Fourier Transform
VAD	Voice Activation Detection
ZCR	Zero Crossing Rate

Chapter 1

Introduction

Speech is the principal mean of communication between human beings. In fact, it is the ability to communicate using language that distinguishes humans from other living species. Speech being the language conveyor has always fascinated researchers. The profusion of multimedia in our century provides great incentives for research in speech communication. It is much easier to give an oral command to a machine than typing a sequence of characters on a keyboard. Nowadays, we can b uy dictation software like IBM's ViaVoice [26] or Nuance's Dragon Software [63] which are very efficient. We can also find voice based control in most modern telephones and even cars can be voice controlled.

Actually, speech recognition has started at the beginning of last century. The first known example of a speech recognition machine is a toy called "Radio Rex" [21] invented in 1922. It used mechanical resonance to a frequency of 500 Hz contained in the vowel of the word "Rex" to trigger a response (a dog jumping out from its kennel). In 1938, Dudley [25] invented the "Vocoder" which analyses speech in the frequency domain (using a filter bank) and then reconstructs it. This machine is the first device that provided short time Fourier analysis of the speech signal. The Vocoder is still used by singers to modify their voice. 1946 is the year of short time Fourier analysis. Gabor [37] developed the theory of short time Fourier analysis and the sound spectrograph [52] has been invented the same year.

Figure 1-1 Radio Rex [21]

Figure 1-2 is a spectrogram corresponding to the word "hassen" pronounced by a male speaker [1]. It represents the variation of the power spectrum as a function of time. The intensity is indicated by the grey scale (white represents the smallest value while black the loudest one). The spectrograph allowed speech researchers to have a better view of the dynamic features of the speech signal. We can remark the dark regions in the figure. They represent resonant frequencies ("formants") of the vocal tract.

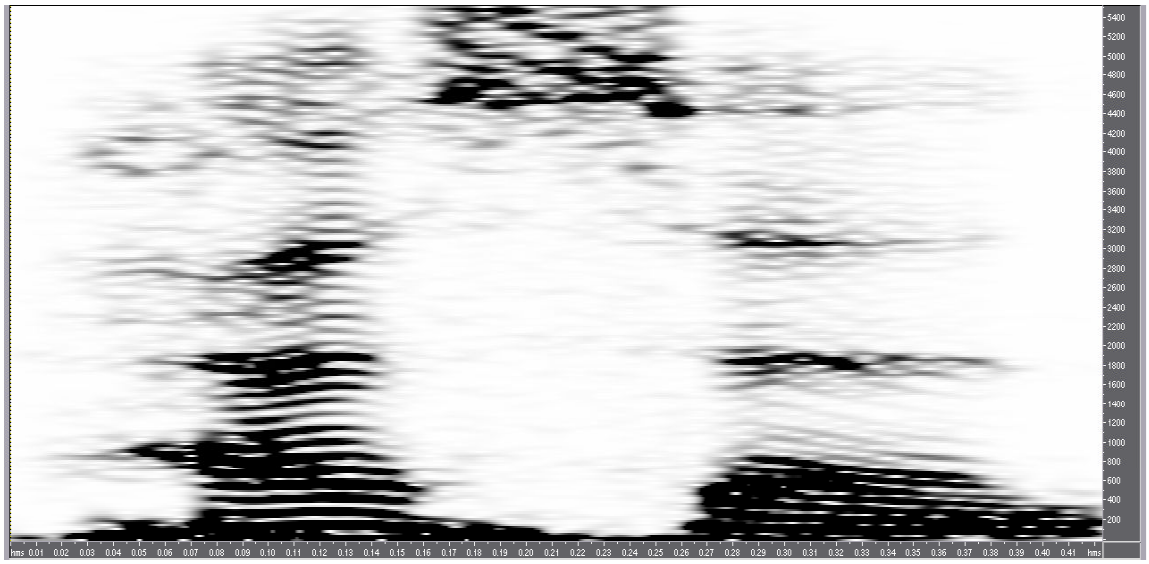


Figure 1-2 Spectrogram of the word "hassen" [1]

The telephone companies pushed researchers to investigate spoken digit recognition in the early 1950's. Most systems were built using analog electronics. The digital computers were still in their infancy. The first electronic digital computer, ENIAC [38], was invented in 1946. We can cite a simple phoneme detector [83] in 1951, the "Audry" spoken digit recognizer [22] in 1952 which achieved a surprising low error rate of 2% and Baumann's word recognizer [9] in 1954.

The development of digital computers in the late 1960's and 1970's allowed the use of sophisticated mathematical methods in the speech area. The speech signal was better modelled using linear prediction coding [3, 44, 45, 55, 58]. Furthermore, "dynamic programming" [10] methods were used to align spoken utterances with stored references by doing a non linear time warping. These different techniques are all grouped under the name of "Dynamic Time Warping" (DTW) [44, 45, 78, 79]. The DTW made speech recognition a reality [71].

During the same period, a revolutionary mathematical tool has been developed by Baum et Al. [5, 6, 7, 8]. Initially called "probabilistic functions of Markov chains", they later came to be called "Hidden Markov Models" (HMM). Their application to speech recognition led to very powerful systems. A leading figure in popularizing HMM's is Lawrence Rabiner [72, 73, 74]. His different articles and books are the main references in the domain. This thesis is part of this continuing endeavour. In our work, we are going to compare classical speech recognition using HMM with a novel method using a "stack algorithm". The idea of using a stack algorithm resulted from the similarity between decoding of a trellis code [90] and the speech recognition process.

If one considers that speech is composed of a discrete sequence of basic units called "*phonemes*[1]", then a model of speech communication system is a discrete one. The source of information is the human brain where the sequence of phonemes is formed, and then the physical means of translation of the above sequence take the relay and convey the information in the shape of a variation of atmospheric pressure. The acoustic wave is picked up by the ear of the listener and translated back as a sequence of phonemes by the listener's brain. An automatic speech recognition system will try to follow the above model as closely as possible.

The adopted model of automatic speech recognition is going to be essentially an encoder (a tree code [96]) that will produce a sequence of "*phones* [2]". The physical mean of transmission and reception of speech is modeled as a channel with discrete input, continuous output as shown in Figure 1-3.

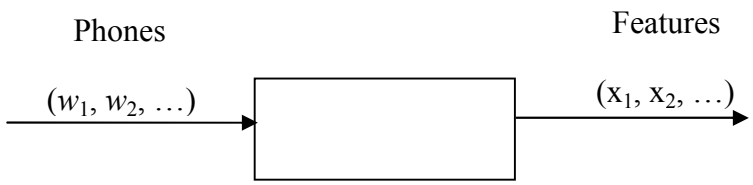


Figure 1-3 Speech Recognition Model

The output of the channel is a sequence of random vectors representing a frame (interval of analysis) of speech such as linear prediction coefficients (LPC parameters), Mel Frequency Cepstral coefficients (MFCC parameters) or any other representation of speech.

[1] This notion will be defined in the next chapter.
[2] A phone is the realization of a phoneme.

In the following section, we give a brief presentation of the thesis. In chapter two, we present the speech signal and develop the different models that will be used in the subsequent chapters. We also present a novel segmentation method of a speech waveform into two groups (plosive, non-plosive). The remaining part of the thesis follows a typical recognition system; starting from signal preprocessing, feature selection, recognition and training.

Chapter three and four are devoted to signal processing prior to feature extraction. Chapter three presents a original method for clipped speech restoration [19] while chapter four is dedicated to noise removal using Wiener filtering. We will show that we can obtain near optimal results using time varying filters.

In chapter five, we describe three popular sets of parametric representation of speech. These parameters are compared statistically. We will show that the MFCC parameters are practically Gaussian distributed and they provide the best separability of classes.

Chapter six is devoted to speech recognition. We are going to describe a recognition system using HMM's for colloquial Algerian Arabic numerals [76]. This scheme will be used as a reference against which we are going to compare our stack algorithm.

Every recognition system requires a training method. We are going to develop an automatic training method based on clustering and a segmentation method based on Viterbi Algorithm in chapter seven.

Finally, we will provide the conclusion and recommendations in chapter eight.

Chapter 2

The Speech Signal

The principal mean of communication between human beings is the language conveyed by the speech signal. This signal is highly structured and it is this structure that transmits information. Unfortunately, this implies that the speech signal is a non-stationary random process. However, we are going to see that we can model this signal as a sequence of quasi-stationary waveforms.

2.1 A Production Model of the Speech Signal

The acoustic waveform is a longitudinal vibration (variation of the atmospheric pressure) produced by a flow of air expulsed from the lungs. Figure 2-1 shows a cross section of the human speech organ. The lungs with the diaphragm are the main source of energy in the speech production mechanism.

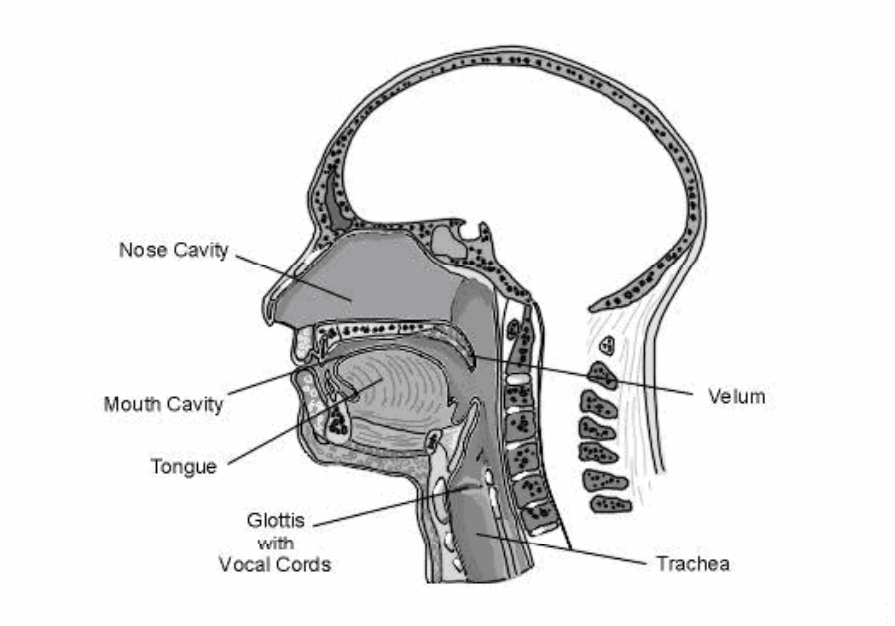


Figure 2-1 The Human Speech Organ [36]

This mechanism of speech production is well described by Flanagan et Al. [32, 31]. A mechanical model is provided by Figure 2-2. From this model, we can see that speech sounds are produced by the vibration of the vocal chords or by turbulences of air inside the vocal tract or by both mechanisms. In the first case, the speech sound is called

"*voiced*" and it corresponds to most vowels. In the second case, it is called "*unvoiced*" or "*voiceless*" and it corresponds to "*fricative*" like "/s/" if the sound is sustained or to "*unvoiced stops*" like "/p/" if the sound is transient. When both mechanisms are used, we produce sounds like "/z/" called "*voiced fricative*" if they are sustained or "*voiced stops*" like "/d/" for transients.

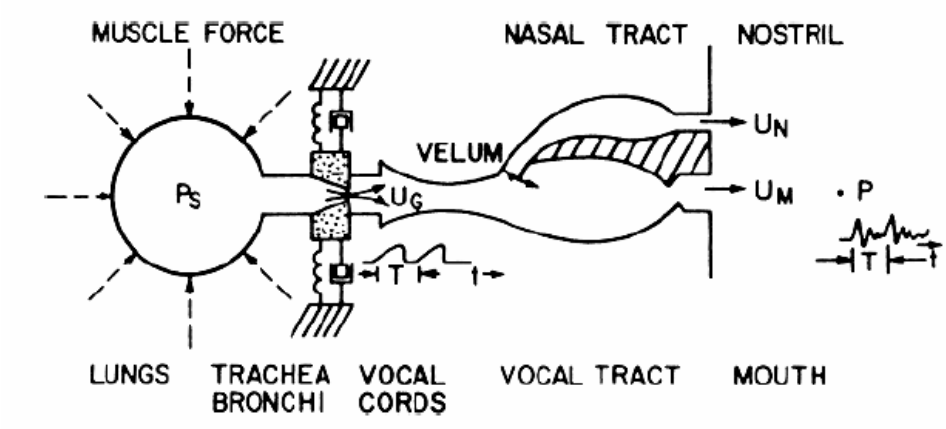


Figure 2-2 Schematic model of the vocal tract system [32].

In the case of voiced speech, the waveform picked by a microphone will contain a quasi-periodic component. The air velocity wave at the glottis (glottal wave) is a rounded saw tooth wave as shown in Figure 2-3. The fundamental frequency of this signal is called the "*pitch*" frequency. The pitch frequency has a value of about 120 Hz for a male speaker and around 220 Hz for a female speaker. Children have even a higher pitch frequency. In the case of unvoiced speech, the waveforms have a noise like nature. These waveforms are going to be modified by the resonances of the vocal tract and the nasal tract.

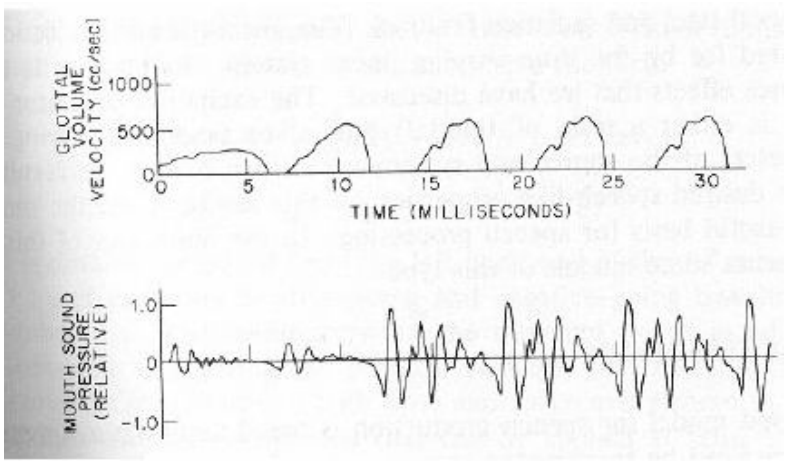


Figure 2-3 Glottal Waveform and Relative Pressure at the Mouth [67]

The basis for most digital speech processing algorithms is the discrete-time model (filter) of Figure 2-4. It represents the process of producing samples of the speech waveform. Such models have become a basis of speech synthesis, speech coding, and speech recognition algorithms. Although the following model is linear, it is fairly accurate.

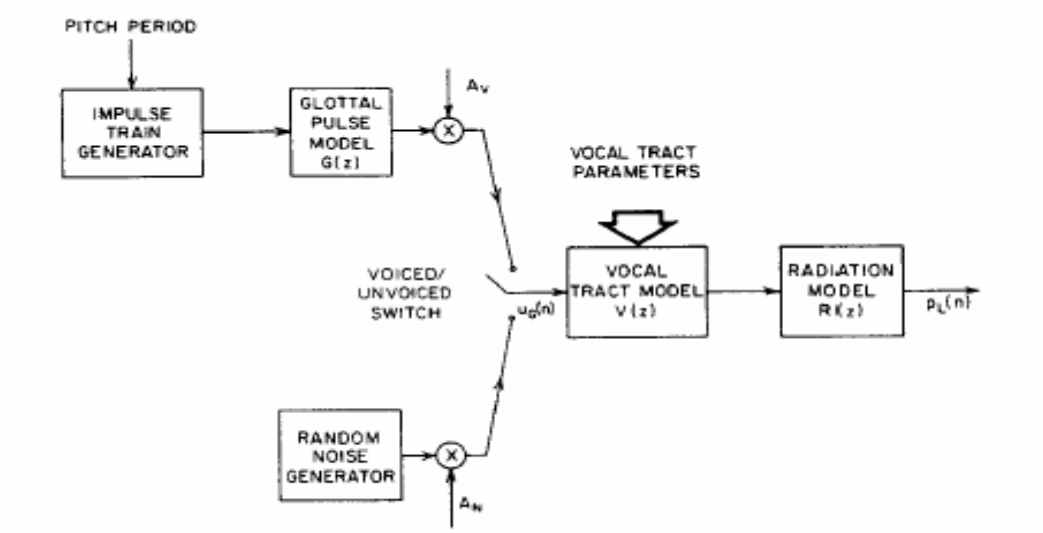


Figure 2-4 Discrete Time Model of Speech Production [70]

2.1.1 Vocal tract model

We have seen that the vocal tract is an acoustic resonator. Its discrete time model is a time varying digital filter $V(z)$. $V(z)$ represents the transfer of the volume velocity between the glottis and the lips. The resonant frequencies of the vocal tract are called "*formants*". These formants correspond to poles of the transfer function $V(z)$. These poles in the s-plane are represented by:

$$s_k, s_k^* = -\sigma_k \pm 2\pi F_k \tag{2.1}$$

The discrete time equivalent poles are:

$$\begin{aligned} z_k, z_k^* &= e^{-\sigma_k T_s} e^{\pm j 2\pi F_k T_s} \\ &= e^{-\sigma_k T_s} \cos 2\pi F_k T_s \pm e^{-\sigma_k T_s} \sin 2\pi F_k T_s \end{aligned} \tag{2.2}$$

In the above equations $2\pi F_k$ represents the formant frequency in rd/s and $2\sigma_k$ represents the formant bandwidth in rd/s. T_s represents the sampling period. The transfer function $V(z)$ can be represented by a cascade of second order resonators:

$$V(z) = \prod_{k=1}^{M} V_k(z) \tag{2.3}$$

where:

$$V_k(z) = \frac{1 - 2|z_k|\cos(2\pi F_k T) + |z_k|^2}{1 - 2|z_k|\cos(2\pi F_k T)z^{-1} + |z_k|^2 z^{-2}} \tag{2.4}$$

The numerator of $V_k(z)$ in equation (2.4) corresponds to a value of $V_k(1) = 1$ ($z = 1$ correspond to zero frequency, $\omega = 0$). The number of formants is usually between three and five. This implies that the order of the filter varies between 6 and 10.

2.1.2 Radiation Model

The pressure at the lips has been shown to be related to the volume velocity at the lips by a transfer function which can be approximated as a first order backward difference [70]. So, the transfer function $R(z)$ is given by:

$$R(z) = R_0\left(1 - z^{-1}\right) \tag{2.5}$$

The above model introduces a zero at dc in the overall transfer function.

2.1.3 Glottal Pulse Model

We have seen in Figure 2-3 (upper waveform) that the glottal volume velocity wave for voiced speech is a quasi-periodic wave composed of a periodic succession of pulses $g(n)$. Rosenberg [77] has analysed this pulse and he found that $g(n)$ can be accurately approximated by:

$$\begin{aligned} g(n) &= \frac{1}{2}\left[1 - \cos\left(\pi n / N_1\right)\right] && 0 \le n \le N_1 \\ &= \cos\left[\pi\left(n - N_1\right) / 2N_2\right] && N_1 \le n \le N_2 \\ &= 0 && \textit{otherwise} \end{aligned} \tag{2.6}$$

$g(n)$ being a finite time pulse has an all zero z-transform. However, the following second order model [58] has provided very good approximation.

$$G(z) = \frac{-ae\ln(a)z^{-1}}{\left(1 - az^{-1}\right)^2} \tag{2.7}$$

2.1.4 The Complete Model

For voiced speech, it is seen that a quite accurate model for speech production is a periodic impulse train filtered by the cascaded filters, $G(z)$, $V(z)$ and $R(z)$. The filter

$V(z)$ is time varying. However, we consider its parameters to be invariant for durations of about 20 ms. So, we can consider a filter $H(z)$.

$$\begin{aligned} H(z) &= G(z)V(z)R(z) \qquad voiced \\ H(z) &= V(z)R(z) \qquad unvoiced \end{aligned} \tag{2.8}$$

The above model does not take into account the effect of the nasal tract which appears when we pronounce sounds like "/m/". The resonances of the nasal tract are going to introduce zeroes in the total transfer function. So, to resume, the complete model is going to be a pole and zero (slowly) time varying filter modifying either a periodic impulse train or a random (white) noise. However, in many cases, speech can be correctly modelled by an all pole model [58, 3].

2.2 Short Time Spectrum Analysis

2.2.1 Definition

We have seen that the transfer function $H(z)$ is a time varying filter. So, this implies that the spectral properties of the speech signal vary with time. So, instead of considering the speech signal as a single entity, we are going to assume that the waveform is a concatenation of many different signals. Each signal has different spectral properties. An example is provided by Figure 2-5. We remark a noise like wave at the beginning of the signal corresponding to the unvoiced sound "/s/". It is followed by quasi-periodic wave with different nature corresponding respectively to the sounds "/a/", "/m/", "/ee/" and finally "/r/".

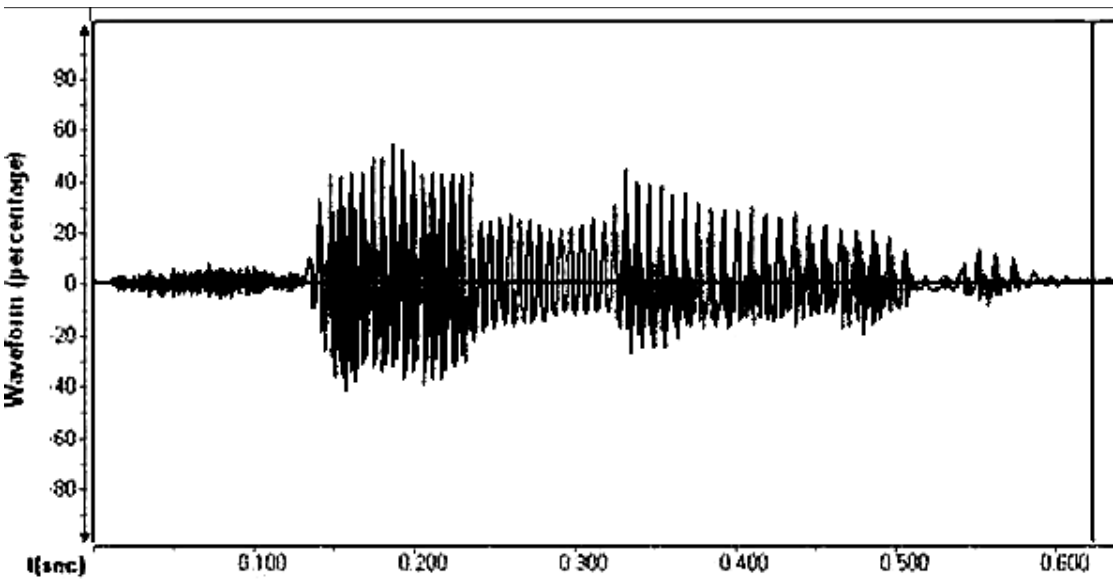


Figure 2-5 Waveform corresponding to the word "samir"

So, quite naturally, a definition [70] of the Short Time Fourier Transform (STFT) is:

$$X_n(e^{j\omega}) = \sum_{m=-\infty}^{+\infty} x(m)\, w(n-m)\, e^{-j\omega m} \tag{2.9}$$

where $x(n)$ is the signal under analysis and $w(n)$ is a finite time window. The above formula is invertible.

$$x(m)\, w(n-m) = \frac{1}{2\pi} \int_0^{2\pi} X_n(e^{j\omega})\, e^{jm\omega} d\omega \tag{2.10}$$

So, if $w(0) \neq 0$, then we can recover $x(n)$ by:

$$x(m) = \frac{1}{2\pi w(0)} \int_0^{2\pi} X_m(e^{j\omega}) e^{j\omega m} d\omega \tag{2.11}$$

Equation (2.9) can be viewed as a convolution:

$$X_n(e^{j\omega}) = \left(x(m) e^{-j\omega m} \right) * w(m) \Big|_{m=n} \tag{2.12}$$

or alternatively:

$$X_n(e^{j\omega}) = \left(x(m) * \left(w(m) e^{j\omega m} \right) e^{-j\omega m} \right) \Big|_{m=n} \tag{2.13}$$

A typical window, like the Hamming window has a lowpass frequency response with a bandwidth being inversely proportional to the window length. So, equation (2.12) corresponds the following sequence of operations: we first shift the frequencies around ω down to zero and then we lowpass filter the resulting signal. The same result is obtained using equation (2.13). $w(m)e^{j\omega m}$ is the impulse response of a bandpass filter centered around the frequency ω. The multiplication by the phasor $e^{-j\omega m}$ shifts the output of the bandpass filter down to zero. So, when we are analyzing a signal using the short time Fourier transform, we have to find a good compromise between a good frequency resolution (long window) and a good time resolution (short window). Figure 2-6 shows the short time spectrum computed at time t = 180 ms using a Hamming window of size = 512 samples (the sampling frequency is f_s = 11025 Hz) for the signal displayed in Figure 2-5. If we change the size of the window to 128 samples, we obtain the spectrum displayed in Figure 2-7. We can remark the complete loss of details.

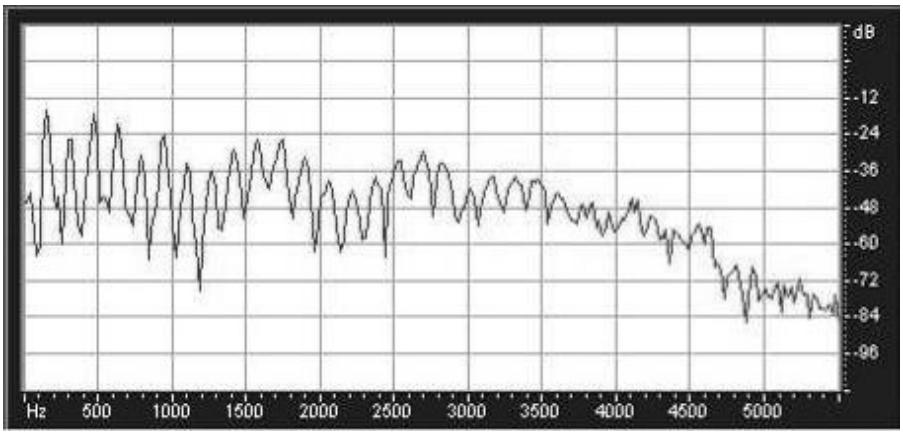


Figure 2-6 Short time power spectrum of vowel "/a/" using a Hamming window of size = 512 samples at time t=180 ms

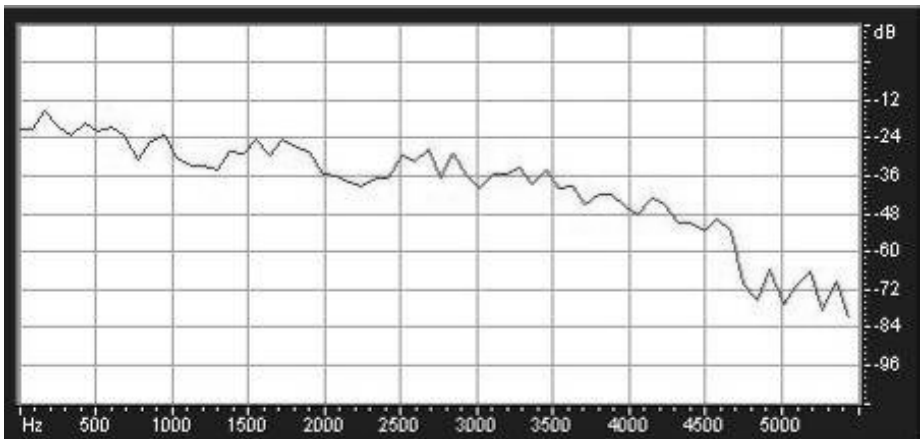


Figure 2-7 Short time power spectrum of vowel "/a/" using a Hamming window of size = 128 samples at time t=180 ms

2.2.2 The Sound Spectrogram

The sound spectrograph was invented in the 1940's [52]. It is an important tool of analysis of speech signals. Up to the 1970's, the sound spectrograph was built using analogue electronic devices. It used a magnetic tape and a variable bandpass filter to produce "spectrograms". A spectrogram is a display of the magnitude of the short time Fourier transform as a function of time. Today, a spectrogram is produced using digital signal processing (windowing followed by fast Fourier transform FFT). In order to generate a spectrogram, we have to use a finite time window and move it by R samples in time and compute the short time Fourier transform on a discrete set of frequencies, i.e., use a discrete Fourier transform (DFT). In other words, we compute:

$$X_{rR}(k) = \sum_{m=rR-L+1}^{rR} x(m)w(rR-m)e^{-j(2\pi k/N)m} \qquad k = 0,1,...,N-1 \qquad (2.14)$$

where N is the number of equally spaced frequencies in the interval $0 \le \omega \le 2\pi$ and L is the window length.

The sound spectrogram is then a plot of the magnitude of X_{rR} expressed in dB.

$$S(t_r, f_k) = 20\log_{10}\left|X_{rR}(k)\right| \qquad (2.15)$$

where the plot axes are labelled in terms of analog time and frequency through the relations:

$$t_r = rRT_s \quad \text{and} \quad f_k = k/NT_s \qquad (2.16)$$

where T_s is the sampling period of the signal $x(n)$. So, a spectrogram is a plot of a function of two variables: time and frequency. The magnitude is displayed using either a grey scale or a color scale. Depending on the window size, the spectrogram is called wide band (short window) or narrow band (long window).

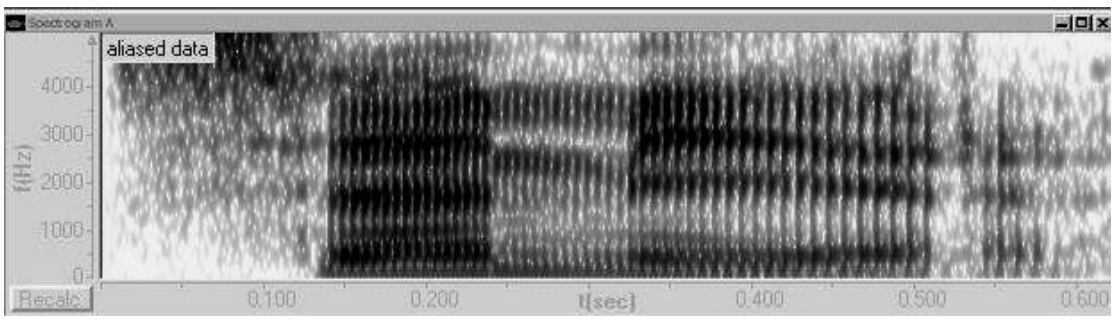


Figure 2-8 Wide band spectrogram of the word "samir"

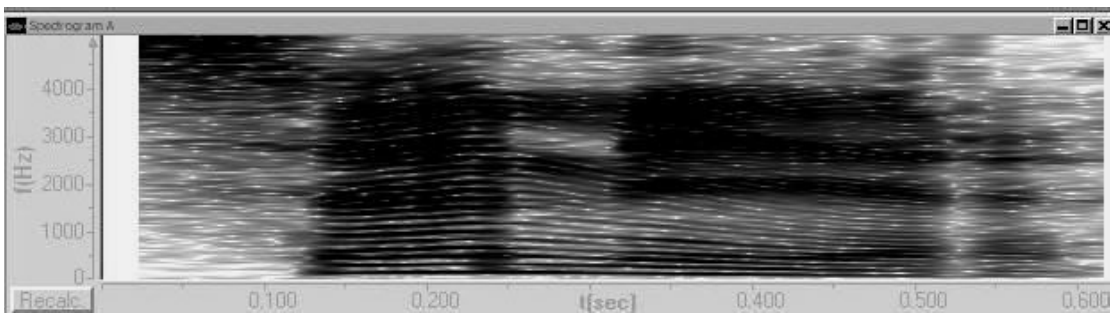


Figure 2-9 Narrow band spectrogram of the word "samir"

In Figure 2-8 and Figure 2-9, we see two different spectrograms for the same speech signal. In the wide band, we can observe very clearly the formants characterizing the voiced part of the signal. We can also notice quite well the time variation of the

spectrum. In the narrow band, the modulation of the spectrum by the pitch frequency is apparent; however, the boundaries of the word segments become fuzzier.

2.3 Phonetics and Phonology

Phonology is a subfield of linguistics which studies the sound system of a specific language or group of languages [41, 47, 87, 89]. In phonology, we define basic sounds that are distinct units of a given language or a group of languages. This "atomistic" view of speech is commonly used in most automatic speech recognition (ASR) systems. We view words as a succession or concatenation of basic units called "*phonemes*". Phonemes are language dependent and are abstraction of basic sounds called "*phones*". For example, the phoneme /t/ in "ten" and in "that" represents the same entity, however, the two instances sound differently. They are called "*allophones*". So, in general, it is the phones that are used as a basic unit in ASR systems. For the English language, we can use the set of phonemes given by the ARPAbet [3] set shown in Table 2-1. This table can be freely downloaded from the Carnegie Mellon University site: http://www.speech.cs.cmu.edu/cgi-bin/cmudict. Another phonetic alphabet is the International Phonetician Association alphabet (IPA) [40, 2], which is used in almost all dictionaries to indicate the pronunciation of the different words. The IPA alphabet is composed of 130 different symbols and it is possible to phonetically encode practically all the languages with the IPA symbol set. However, it uses a special set of graphical symbols, which makes it difficult to use in practice. For the Arabic language, Harakat [41] proposes a list of 31 phonemes. This list is provided in Table A-1 and will be used in our work. Another list of phonemes for the Arabic language is proposed by Reggab [76]. It will be used occasionally in our work. In what follows, we provide a short description of the phonemes listed in Table A-1.

Vowels

Vowels (/aa/, /ii/, /oo/) are voiced sound with rather fixed parameters (formants). They can be sustained for a quite long time.

[3] ARPA, (now DARPA) stands for the Advanced Research Projects Agency (The D stands for Defense): It is a United State Government Agency which is responsible for the funding of many research projects in speech recognition.

Semivowels

This group of sounds consisting of /wa/ and /ya/ is quite difficult to characterize. These sounds are called semivowels because of their vowel like nature. They are characterized by a gliding transition in the vocal tract area between adjacent phonemes.

Class	ARPAbet	Example	Transcription
Vowels and diphthongs	IY	b*ee*t	[B IY T]
	IH	b*i*t	[B IH T]
	EY	b*ai*t	[B EY T]
	EH	b*e*t	[B EH T]
	AE	b*a*t	[B AE T]
	AA	b*o*b	[B AA B]
	AO	b*or*n	[B AO R N]
	UH	b*oo*k	[B UH K]
	OW	b*oa*t	[B OW T]
	UW	b*oo*t	[B UW T]
	AH	b*u*t	[B AH T]
	ER	b*ir*d	[B ER D]
	AY	b*uy*	[B AY]
	AW	d*ow*n	[D AW N]
	OY	b*oy*	[B OY]
Glides	Y	*y*ou	[Y UH]
	R	*r*ent	[R EH N T]
Liquids	W	*w*it	[W IH T]
	L	*l*et	[L EH T]
Nasals	M	*m*et	[M EH T]
	N	*n*et	[N EH T]
	NG	si*ng*	[S IH NG]
Stops	P	*p*at	[P AE T]
	B	*b*et	[B EH T]
	T	*t*en	[T EH N]
	D	*d*ebt	[D EH T]
	K	*k*it	[K IH T]
	G	*g*et	[G EH T]
Fricatives	HH	*h*at	[HH AE T]
	F	*f*at	[F AE T]
	V	*v*at	[V AE T]
	TH	*th*ing	[TH IH NG]
	DH	*th*at	[DH AE T]
	S	*s*at	[S AE T]
	Z	*z*oo	[Z UW]
	SH	*sh*ut	[SH AH T]
	ZH	a*z*ure	[AE ZH ER]
Affricates	CH	*ch*ase	[CH EY S]
	JH	*j*udge	[JH AH JH]

Table 2-1 Condensed list of ARPAbet phonetic symbols for North American English

Nasals

The nasal consonants, /mm/ and /nn/, are voiced sounds characterized by the fact that the velum is lowered during their pronunciation. Air flows through the nasal tract. They are characterized by zeroes in the transfer function $H(z)$.

Unvoiced Fricatives

As the name indicates, the unvoiced fricatives /ff/, /st/, /ss/ and /sh/ produce noise-like waveform. They are produced by air turbulences inside the vocal tract and there is no glottal excitation.

Voiced Fricatives

The voiced fricatives, /th/, /zz/ and /dh/ are the counterpart of the unvoiced ones. They are produced by the same constriction in the vocal tract. The difference is that the vocal cords are used in their production in addition to the air flow turbulence produced by the constriction.

Voiced Stops (Plosives)

The voiced stop consonants /bb/, /dd/, /dj/, and /da/, are transient, non continuant sounds which are produced by building up pressure behind a total constriction somewhere in the oral tract, and suddenly releasing the pressure. For /bb/ the constriction is at the lips; for /dd/ the constriction is back of the teeth. During the period when there is a total constriction in the tract there is no sound radiated from the lips. However, there is often a small amount of low frequency energy radiated through the walls of the throat (sometimes called a voice bar). This occurs when the vocal cords are able to vibrate even though the vocal tract is closed at some point.

Unvoiced Stops (Plosives)

The unvoiced stop consonants /ta/, /tt/, /qa/, /kk/ and /ia/ are similar to the voiced ones with one major exception. During the period of total closure of the tract, as the pressure builds up, the vocal cords do not vibrate. Thus, following the period of closure, as the air pressure is released, there is a brief interval of friction (due to sudden turbulence of the escaping air) followed by a period of aspiration (steady air flow from the glottis exciting the resonances of the vocal tract) before voiced excitation begins.

Figure 2-10 shows the major four categories of phonemes that are of concern in our research. These four groups will be collected later into two ensembles: "*sustainable*" and "*non-sustainable*" phonemes.

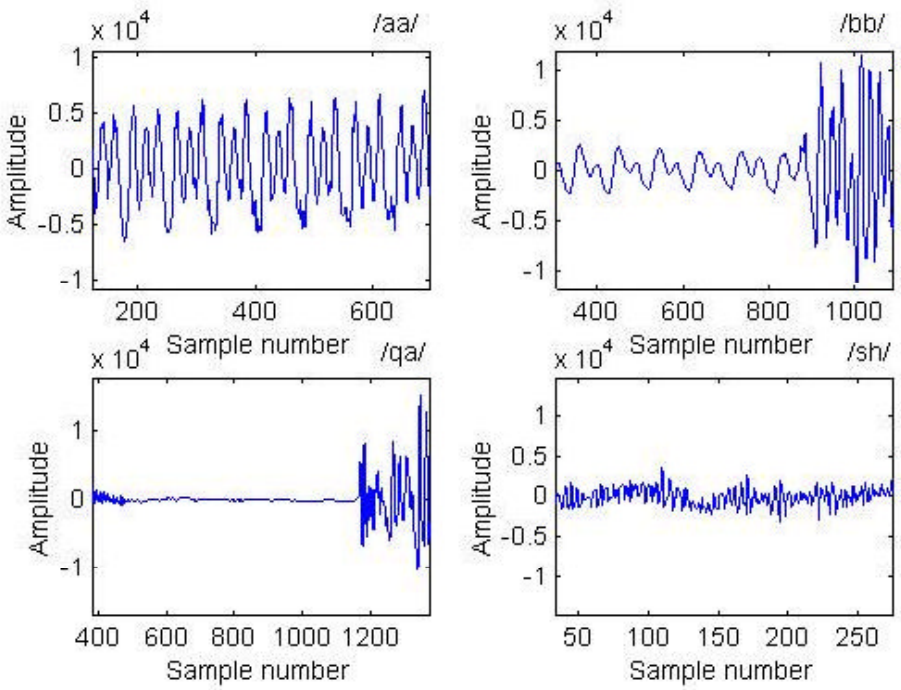


Figure 2-10 Acoustic waveforms of the four major types of phonemes

2.4 Short Time Characterization of Speech

One of the first steps in speech processing is to decide whether a waveform segment corresponds to speech or to noise (silence). This operation is usually accomplished using a voice activation detection (VAD) algorithm [75, 62]. Most of the

VAD algorithms use the concept of short time energy (or magnitude) and zero crossing rate (ZCR).

Using the same notation as in section 2.2.2, consider a section of length L of signal $x(k)$ positioned at the n^{th} sample, we define the short time energy function [70] as:

$$E_n = \sum_{m=-\infty}^{\infty} \left(x(m)\right)^2 w(n-m) \tag{2.17}$$

where $w(n)$ is a window of size L. E_n as given by equation 2.17 is more expensive to compute than the mean magnitude function:

$$M_n = \sum_{m=-\infty}^{\infty} \left|x(m)\right| w(n-m) \tag{2.18}$$

A large value of E_n or M_n is a strong indicator of the presence of a voiced segment in the signal. However, a small value does not imply that the actual segment is a silence segment. Unvoiced segments have a quite low value of energy or magnitude. We have seen that they are noise-like waveforms. Their spectrograms show that the energy is concentrated at high frequency. A simple parameter characterizing this behaviour is the zero crossing rate:

$$ZCR_n = \sum_{m=-\infty}^{+\infty} \left|\text{sgn}[x(m)] - \text{sgn}[x(m-1)]\right| w(n-m) \tag{2.19}$$

In many cases, the window is a rectangular window.

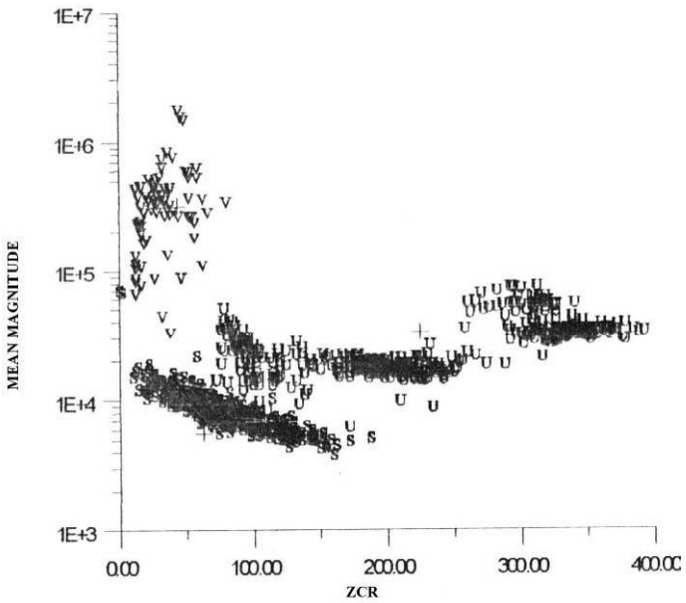


Figure 2-11 Mean magnitude and ZCR scatter plot

We have used M_n and ZCR_n to segment acquired speech signals into voiced, unvoiced and silent segments. Figure 2-11 shows a scatter plot for 406 different speech segments from a male locutor. The letter "v" indicates the segment is voiced, the letter "u" indicates that it is unvoiced and "s" indicates a silent segment. It shows clearly how to discriminate between the different segments. A silent segment is characterized by a low M_n and a low ZCR_n. A voiced segment on the other hand has a quite high M_n but a low ZCR_n. Finally, an unvoiced segment has a low M_n and a high ZCR_n. Table 2-2 shows the result of classifying 718 segments of speech into voiced, unvoiced and silent segments.

Classified as / Input Segment	Voiced	Unvoiced	Silence
Voiced	99.47%	0.52%	0.01%
Unvoiced	0.0%	99.8%	0.2%
Silence	0.9%	0.0%	99.1%

Table 2-2 Confusion Matrix (Voiced, Unvoiced, Silence)

Another segmentation that can be performed on a given speech waveform is the identification of plosives [15, 61]. We can see from Figure 2-10 that plosives are purely dynamic in nature. A plosive is defined essentially by the sudden variation of amplitude that follows the closure of the mouth. The criterion that we have used is the ratio of the magnitude function between successive frames. This ratio seems to be a natural candidate for plosive detection since a plosive is characterized by a complete closure ($M_{n-1} \approx 0$) followed by sudden release of energy (M_n quite large).

However, the use of the sole magnitude ratio as a criterion is not enough because it also detects the transitions between weak unvoiced sounds (/ff/ for example) and a voiced sound like a vowel. We added a measure of ZCR before the transition in order to eliminate the above mentioned problem. If the ZCR happens to have a high value, then it is evident that we have an unvoiced/voiced transition and not a plosive.

The plot shown in Figure 2-12 is the acoustic signal of the word “FAROUK” ‘فاروق’, where we have the weak unvoiced sound /ff/ at the beginning followed by the vowel /aa/. In such a case, if the magnitude ratio is the only parameter used in the detection of plosives, a plosive will be detected around sample 500 where in fact it is

not. When the ZCR information was not used, the system detected two plosives, one around sample 500, which is a wrong decision, and another one around sample 7500 where the plosive /qa/ does really exist. However, after the use of the ZCR information, the first decision was eliminated due to the high ZCR of the sound /ff/. A test performed on 50 words of the same type provided 91% of correct results.

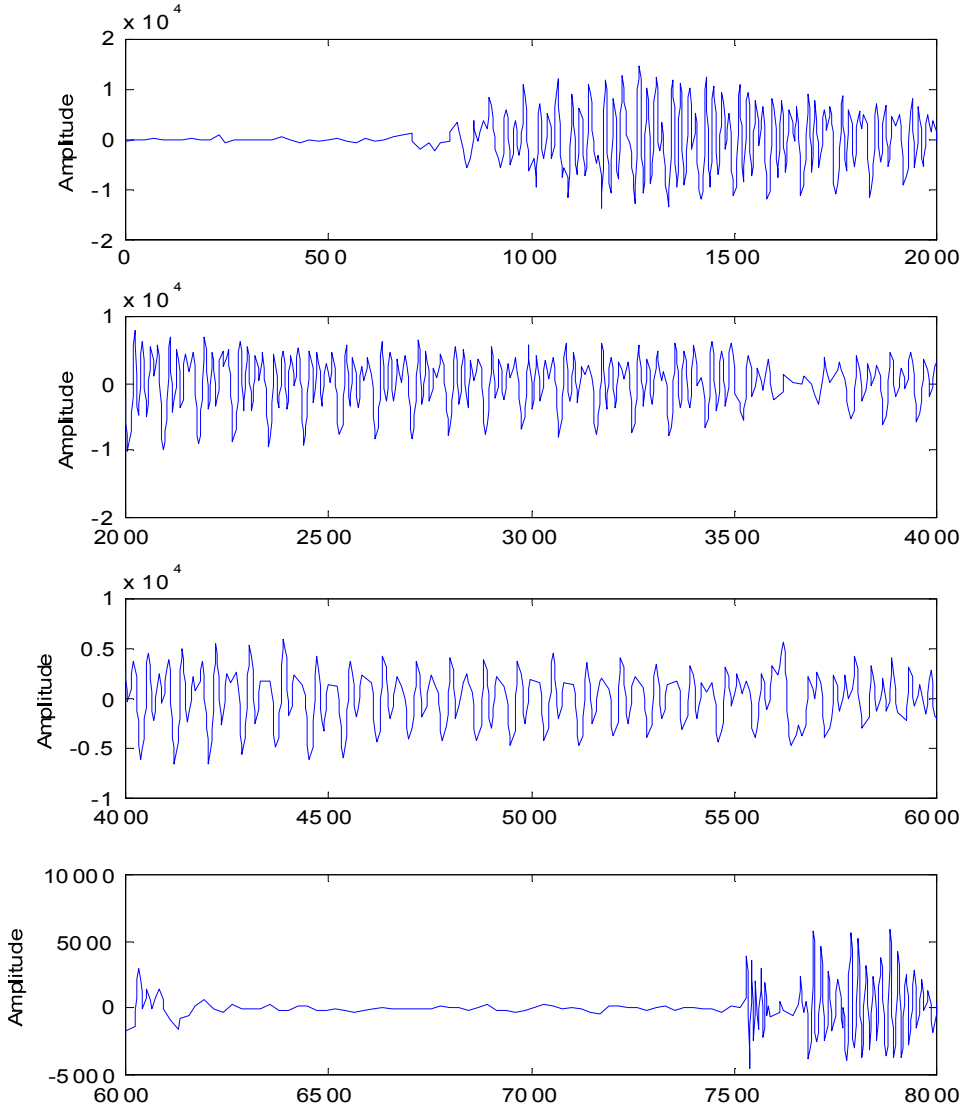


Figure 2-12 The acoustic signal of the word 'فـــاروق'

Finally, we have seen that a speech signal is composed of a concatenation of phones. Some of them can be sustained while others are purely transient. The average duration of a phone is about 20 ms and it is known that the maximum duration of a phone cannot exceed 100 ms [31, 70]. From this, we can conclude that speech can be considered as a quasi-stationary signal if the interval of observation does not exceed 20 ms.

Chapter 3

Clipped Speech Restoration

The first stage in a speech recognition system is the acquisition system. If the ASR system is implemented on a personal computer (PC), this acquisition system is the PC sound card. The signal is sampled at a rate that can be adjusted between 6 kHz and 119 kHz for most sound cards, but usually it does not exceed 44.1 kHz (sampling period larger than 22.73 μs). The analog to digital converter (ADC) represents the speech samples with 16 bits. The signal is picked by a low cost microphone and in a quite noisy environment (without the use of a sound anechoic chamber). A PC sound card does not have an automatic gain control (AGC) system. This means that the acquired signal is confronted with two major problems: DC level wandering and peak clipping. The first one is easily eliminated by simple linear processing but the second one requires more complex algorithms. Peak clipping is fundamentally a non linear distortion. It is characterized by the fact that several successive values of the signal disappear and are replaced by a constant. However, we have seen in chapter 2 that the speech signal is highly predictable. Therefore, clipped speech restoration is an interpolation problem. We are going to use the signal known values to predict the missing samples [19].

3.1 Basic Interpolation Methods

When there is no a priori information on the signal, classical numerical interpolation methods should be used [43, 69, 103]. The classical polynomial interpolation methods are based on Weierstrass approximation theorem [43]. These methods include the Lagrange polynomials, the divided difference polynomials (Newton), etc. These methods assume that the data is not noisy and are not well suited to the problem at hand. A smoother approximation is provided by cubic splines [43, 69, 103]. However, even here, noise is a problem in the interpolation and furthermore, we are not using the known signal properties for interpolation.

The classical bandlimited interpolation methods [18] use Shannon's interpolation formulation of the sampling theorem [82]. They are commonly used for sampling rate up conversion. To increase the sampling rate by I (integer), we insert $I-1$ zeroes between samples and low pass filter the obtained signal. Figure 3-1 shows the result of

sampling rate change on clipped speech (from 16 kHz to 44.1 kHz). We see clearly that the clipping is not corrected. Furthermore, the clipped part of the signal, which was horizontal in the original signal, contains ripples due to the "sinc" function of the lowpass interpolating filter.

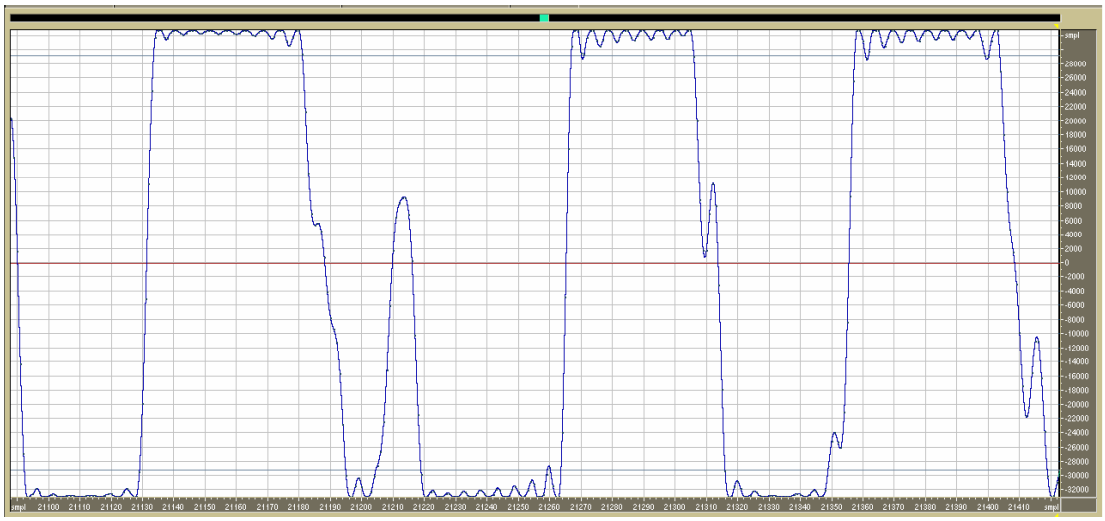


Figure 3-1 Result of sampling rate increase on clipped speech

Consequently, the above listed methods of interpolation are not suited to the problem at hand. This has led us to use statistical model based interpolation [92] for computing the missing values.

3.2 Justification of the Method

The statistical model based method used to eliminate peak clipping in the speech signal is based on linear prediction [3, 44, 45, 55, 58, 70]. It uses the fact that speech signal is highly predictable. As seen in chapter 2, a speech segment is composed of a sequence of voiced, unvoiced and silent (noise) segments. The type of speech signal that has the greatest probability for being peak clipped is voiced speech [70]. Figure 2-11 represents a scatter plot of voiced, unvoiced and silence mean magnitude and zero crossing rate of segments of speech. Voiced speech segments are indicated by the letter "V", unvoiced segments by the letter "U" and the silent segments by the letter "S". It shows clearly that the voiced signals cluster at high mean magnitude values.

Fortunately, voiced speech happens to be quite predictable. We have described the production model in chapter 2. From equation (2.8), it is clear that voiced speech follows quite closely the linear prediction equations. Commercial software like DC-6, from Diamond Cut products, use low order linear prediction for clipped audio signal restoration and the problem of audio signal interpolation have also been addressed by

Vaseghi [92] who uses linear prediction from adjacent samples and samples one period away (audio signals are assumed to be periodic).

Voiced speech can be considered as a quasi periodic signal. It can be modelled as the output of a linear time invariant system (during few milliseconds, the system can safely be assumed to be time invariant) driven by a periodic train of impulses. In this case, a quite general formulation of the speech signal x_n will be[4]:

$$x_n = \sum_{k=1}^{p} a_k x_{n-k} + \sum_{k=0}^{p} b_k u_{n-k} \tag{3.1}$$

where u_k is equal to 1 every T seconds and zero otherwise, T being the pitch period. a_k and b_k are respectively the recursive and non recursive parameters of the above production filter $H(z)$ of order p. So, within a pitch period (N_T samples) and after p samples, we can write:

$$x_n = \sum_{k=1}^{p} a_k x_{n-k} \tag{3.2}$$

The above equation breaks down in the part of the speech signal that is clipped. So, if we start the time axis at the beginning of a pitch period and if we call N_T the number of samples within the pitch period, we can write:

$$\begin{aligned} & x_n = \sum_{k=1}^{p} a_k x_{n-k} \quad ; \; p \le n \le N_T \\ & for \quad |x_n| < X_{\max} \end{aligned} \tag{3.3}$$

$X_{\max}$ being the saturation value.

3.3 The Proposed Restoration Algorithm

The proposed algorithm for clipped speech restoration is going to be based on linearly predicting the missing values using equation (3.2). So, the algorithm consists of two following steps:

- Computation of the prediction coefficients a_k .

- Linear prediction of the missing values.

[4] We use the notation x_n instead of $x(n)$ in this chapter because it is more commode for the Kalman recursion.

3.3.1 Computation of the prediction coefficients

The computation of the prediction coefficients a_k can be accomplished either by using a least square solution or by using a recursive algorithm based on Kalman filtering.

For the least square algorithm, we can use equation (3.3) and build the following matrix vector equation relating speech samples x_k:

$$\begin{pmatrix} x_{p+1} \\ x_{p+2} \\ . \\ . \\ x_{N_T} \end{pmatrix} = \begin{pmatrix} x_p & x_{p-1} & . & . & x_1 \\ x_{p+1} & x_p & . & . & x_2 \\ . & . & . & . & . \\ . & . & . & . & . \\ x_{N_T-1} & x_{N_T-2} & . & . & x_{N_T-p} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ . \\ . \\ a_p \end{pmatrix} \tag{3.4}$$

in which all the rows such that $|x_k| = X_{\max}$ are deleted. Equation (3.4) can be written as:

$$\mathbf{b} = \mathbf{X}\ \mathbf{a} \tag{3.5}$$

and the least square solution of equation (3.5) can be obtained as [54]:

$$\mathbf{a} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{b} \tag{3.6}$$

Another approach to the evaluation of the prediction coefficient is the following sequential algorithm (Kalman filter) [39, 84] based on the subsequent set of equations and on an autoregressive model. Consider the next state equation:

$$\mathbf{a}(k+1) = \mathbf{a}(k) + \mathbf{w}(k) \tag{3.7}$$

where $\mathbf{a}(k) = \left(a_1, a_2, \ldots, a_p\right)^T$ and $\mathbf{w}(k)$ is a white stationary sequence. Our problem is to estimate **a**(k) on the basis of observations x_1, x_2, … [84]. The observation model is given by:

$$x_n = \sum_{i=1}^{p} a_i\, x_{n-i} + b_0\, u_n \tag{3.8}$$

and let us consider $\mathbf{C}(k\text{-}1) = [x_{k\text{-}1}, x_{k\text{-}2}, \ldots, x_{k\text{-}p}]$. The observation model becomes:

$$z(k) = x_k = \mathbf{C}(k-1)\mathbf{a}(k) + v(k) \tag{3.9}$$

if it is taken that: $v(k) = b_0 u_k$.

then, starting from an initial estimate $\hat{\mathbf{a}}(0)$, we obtain the following recursion:

$$\hat{\mathbf{a}}(k+1) = \hat{\mathbf{a}}(k) + \mathbf{K}(k+1)[x_{k+1} - \mathbf{C}(k)\hat{\mathbf{a}}(k)] \tag{3.10}$$

where **K** is the Kalman gain given by:

$$\mathbf{K}(k+1)=\frac{\mathbf{V}_{\hat{\mathbf{a}}}(k)\mathbf{C}^{\mathrm{T}}(k)}{b_0^2+\mathbf{C}(k)\mathbf{V}_{\hat{\mathbf{a}}}(k)\mathbf{C}^{\mathrm{T}}(k)} \tag{3.11}$$

and the matrix $\mathbf{V}_{\hat{\mathbf{a}}}$ is the variance matrix of the estimator $\hat{\mathbf{a}}$ and is given by the following equation:

$$\mathbf{V}_{\hat{\mathbf{a}}}(k+1)=[\mathbf{I}-\mathbf{K}(k+1)\mathbf{C}(k)]\mathbf{V}_{\hat{\mathbf{a}}}(k)+\mathbf{V}_{\mathbf{w}} \tag{3.12}$$

where $\mathbf{V}_{\mathbf{w}}$ is the variance matrix of the white noise process $\mathbf{w}(k)$.

The algorithm can be initialized by: $\mathbf{V}_{\mathbf{w}}=\sigma^2\mathbf{I}$, $\mathbf{V}_{\hat{\mathbf{a}}}(0)=\mathbf{0}$ and $b_0=1$ and stopped by using the criterion:

$$\left\|\hat{\mathbf{a}}(k+1)-\hat{\mathbf{a}}(k)\right\|^2\le\varepsilon \tag{3.13}$$

The stopping criterion can also be used for pitch detection because it is evident that the above norm will be large while being in a clipped part, since the autoregressive model will not be valid.

3.3.2 Interpolation of the missing samples

For the computation of the missing samples, equation (3.2) can be used starting from *p* previous samples. This interpolation is referred to as forward. The missing samples can also be predicted from *p* samples that follow the missing part. The first sample can be obtained by solving equation (3.2) as:

$$x_{n-p}=\sum_{i=1}^{p}\alpha_i x_{n-p+i} \qquad ; \quad p+1\le n\le N_T \tag{3.14}$$

where the coefficients α_i are computed from the coefficients a_i using:

$$\alpha_1=-\frac{a_{p-1}}{a_p}\ ;\ \alpha_2=-\frac{a_{p-2}}{a_p}\ ;\ \cdots\ ;\ \alpha_{p-1}=-\frac{a_1}{a_p}\ ;\ \alpha_p=\frac{1}{a_p} \tag{3.15}$$

Consequently, the reconstruction is done using backward interpolation.

3.4 Results

In order to test the previously defined algorithms, we are going to use synthetic and natural speech. The natural speech comes from a very large database of speech samples that were collected for the construction of a speech recognition system in colloquial Algerian Arabic [76]. The pitch frequency is about 100 Hz for male speaker and about 220 Hz for a female one. This corresponds to a pitch period *T* being

between 4.5 ms to 10 ms. So, if a reliable estimation of the prediction parameters is desired, we need a fairly high sampling frequency. For example, a sampling frequency of 10 kHz (sampling period of 100 μs) will provide between 45 and 100 samples for a pitch period. A sampling frequency of 44.1 kHz (sampling period of 22.73 μs) is chosen, which provides between 198 and 440 samples for a pitch period, which is quite reasonable. Also, in all of the following tests, the speech signal is normalized to a maximum value of one.

3.4.1 Synthetic Speech

The algorithm is tested first with a synthetic vowel. The choice of synthetic speech is motivated by the fact that it follows exactly the linear prediction model. The vowel /a/ is generated using the following formants [17]:

FORMANT	$F_i(Hz)$	$BW_i(Hz)$
1	730	60
2	1090	100
3	2440	120
4	3500	175
5	4500	281

Table 3-1 Formants Frequencies and Bandwidth [17]

- The frequencies (F_i) and the bandwidths (BW_i) necessary to specify each formant are shown in the following table.

- The pitch frequency is 120 Hz (male speaker), which corresponds to $N_T = 367$ samples.

Figure 3-2 displays a Z domain pole and zero plot of the obtained transfer function $H(z)$. Figure 3-3 shows few periods of the synthetic vowel /aa/. The prediction order is set to $p = 10$. This signal is clipped to a level of ±0.5 and restored using both methods (least square and Kalman filter method).Figure 3-4 shows one pitch period of the clipped signal. A window of at least 75 samples following the clipped region is used

to compute the predictor coefficients. The first reconstruction is done using the least square estimation of the prediction coefficients.

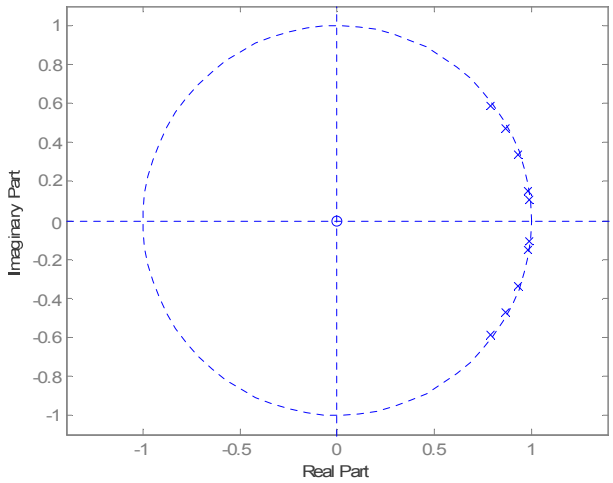


Figure 3-2 Pole and Zero Plot of $H(z)$

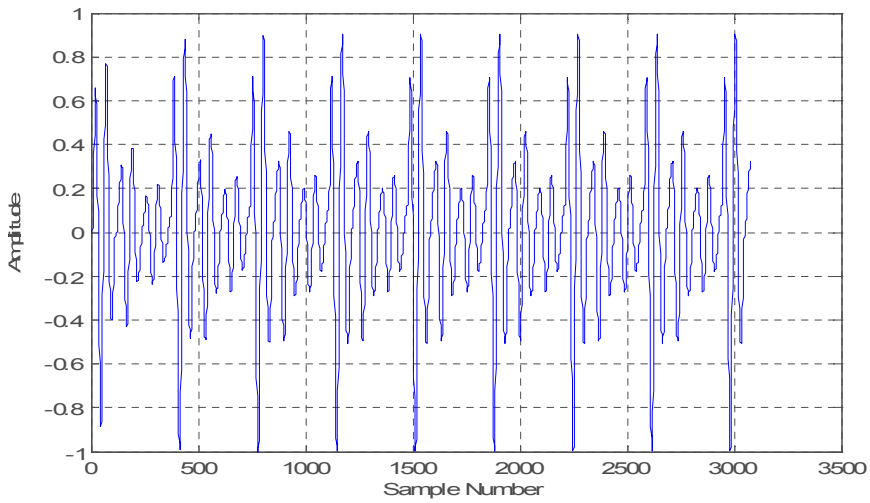


Figure 3-3 Synthetic Vowel /aa/ Normalized Amplitude Waveform

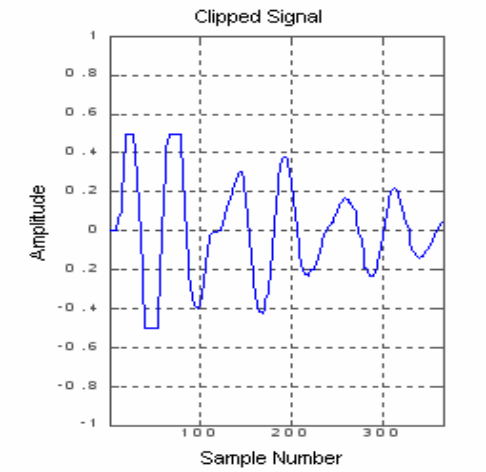


Figure 3-4 Clipped Artificial Vowel

The least square computation of the prediction coefficients along with both forward and backward reconstruction produces the error plots shown in Figure 3- 5. It can be seen that the backward error is much smaller than the forward one. Also, the error occurs at the end of the reconstruction. The error can be reduced by performing both reconstructions and averaging the results. However, since the error is essentially a high frequency signal, simple low pass filtering after backward reconstruction yields the same result.

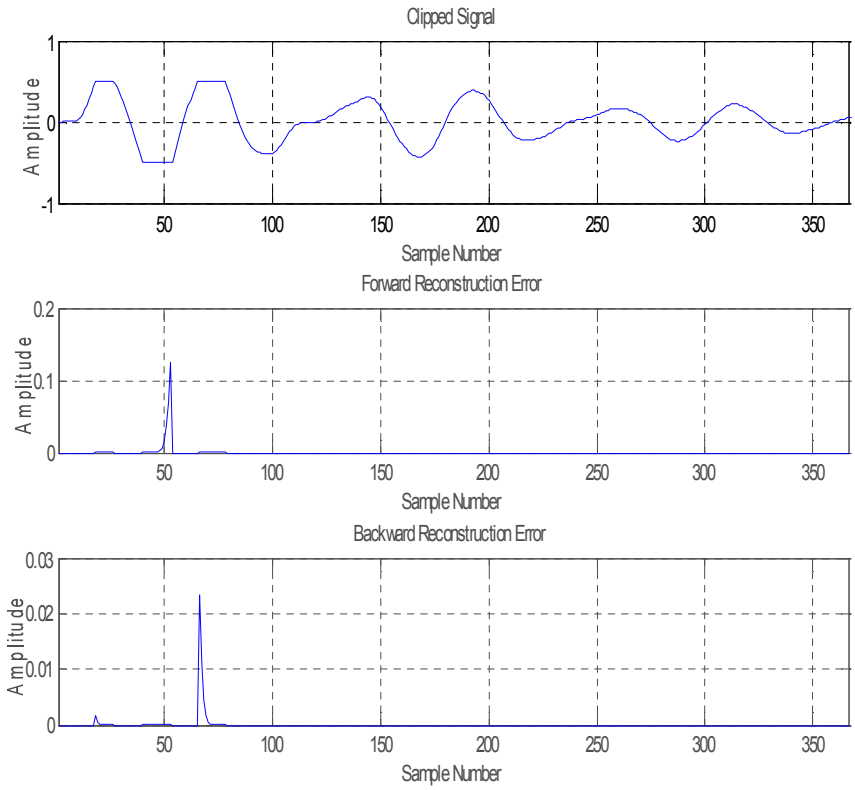


Figure 3- 5 Forward and Backward Reconstruction Error for Synthetic Speech

Kalman filter is also used for the estimation of the prediction parameters. The stopping criterion of the recursive Kalman algorithm is defined as: $\|\mathbf{a}(k+1)-\mathbf{a}(k)\|^2 < \varepsilon$, where ε is a small positive number that describes the convergence of the algorithm. From the plot of $\|\mathbf{a}(k+1)-\mathbf{a}(k)\|^2$ over 04 pitch periods of the signal (under: $\sigma = 0.1 \, and \, b_0 = 1$) shown in Figure 3- 6, it appears that the value ε =0.00008 is acceptable.

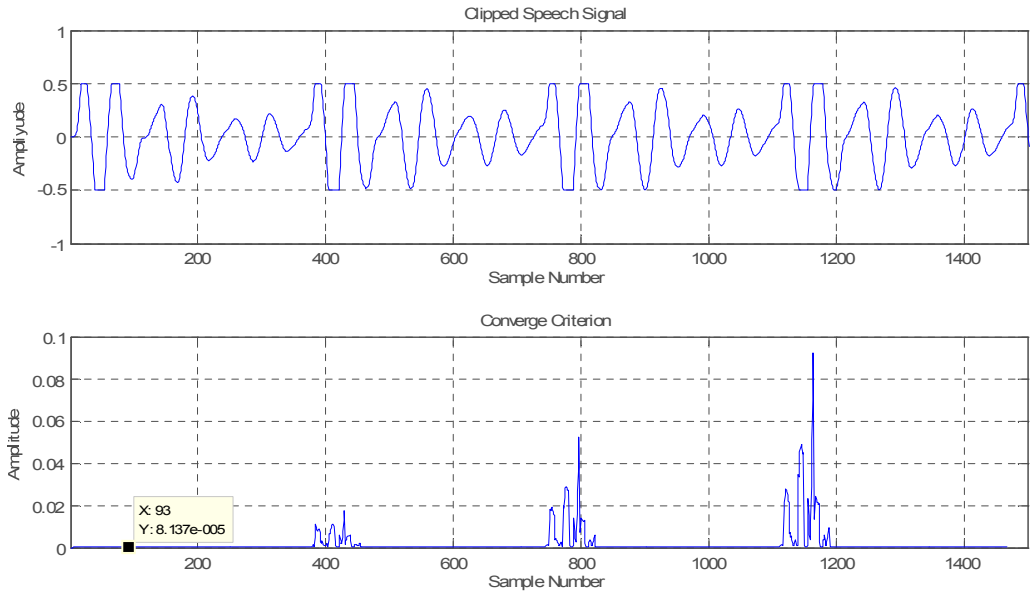


Figure 3- 6 Kalman Filter Convergence Criterion Plot

As stated before, the above convergence criterion ($\|\mathbf{a}(k+1) - \mathbf{a}(k)\|^2$) can be used for pitch detection. This is well illustrated in Figure 3- 6, the large values occurs at the clipped parts generally located at the beginning of the pitch period. After several tests, the following initial values: $\sigma = 0.1$, $b_0 = 1$, $\hat{\mathbf{a}}(0) = \mathbf{0}$. After estimation of the prediction parameters and backward prediction, the error is drawn in Figure 3- 7.

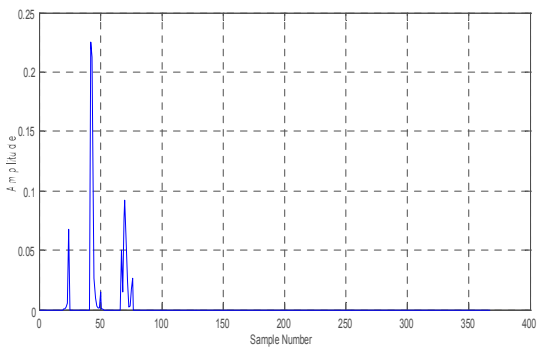


Figure 3- 7 Kalman Filter Error Signal Waveform

We observe that the error using the Kalman filter estimation is much larger that the one using the least square method. Another problem that is encountered is the large computation time. So, in the following tests, the results obtained by the least square method are the only ones that will be presented.

3.4.2 Artificially Clipped Natural Speech

After being applied on a synthetic speech, the proposed technique of interpolation (least square evaluation of the parameters and backward reconstruction) is applied on a voluntarily clipped natural speech. The unclipped signal is taken as a reference when evaluating the reconstruction process precision.

The used recorded speech signal consists on numbers pronounced in Algerian Arabic, sampled at 16 KHz, taken from the database [76]. An audio processing software (Cool Edit Pro 2.1 from Syntrillium Software Corporation) is used to adjust the sampling frequency to 44.1 KHz.

Since the speech is a time varying signal (a concatenation of different sounds with different characteristics) and in order to have a good estimation of the prediction parameters, the following method based on the detection of clipped samples is used: after each detection of a clipped sample, an adjacent segment of enough number of successive unclipped samples (ex.: in our case 75 samples) is considered. If this condition is satisfied, the reconstruction process that uses the least square algorithm for the estimation of the prediction parameters will be applied. Otherwise, the procedure is repeated. Figure 3-8 shows the different steps of signal processing. The original speech and the reconstructed one are practically identical. Figure 3-9 shows the reconstruction error for the natural speech where it can be observed that the error is a high frequency signal with a small peak magnitude. So, a simple low pass filter will eliminate completely the error.

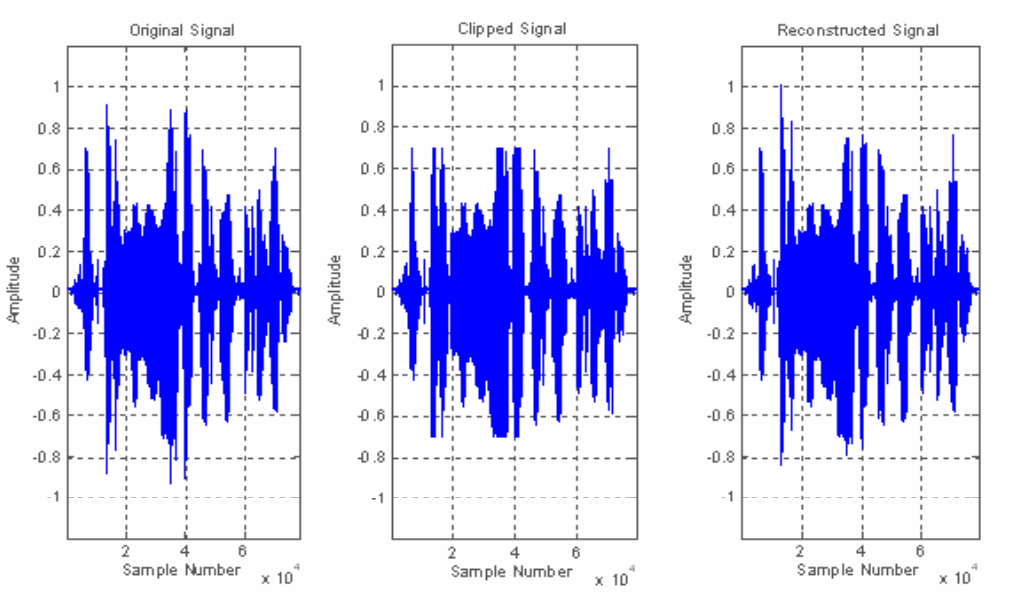


Figure 3-8 Artificially Clipped Natural Speech Reconstruction

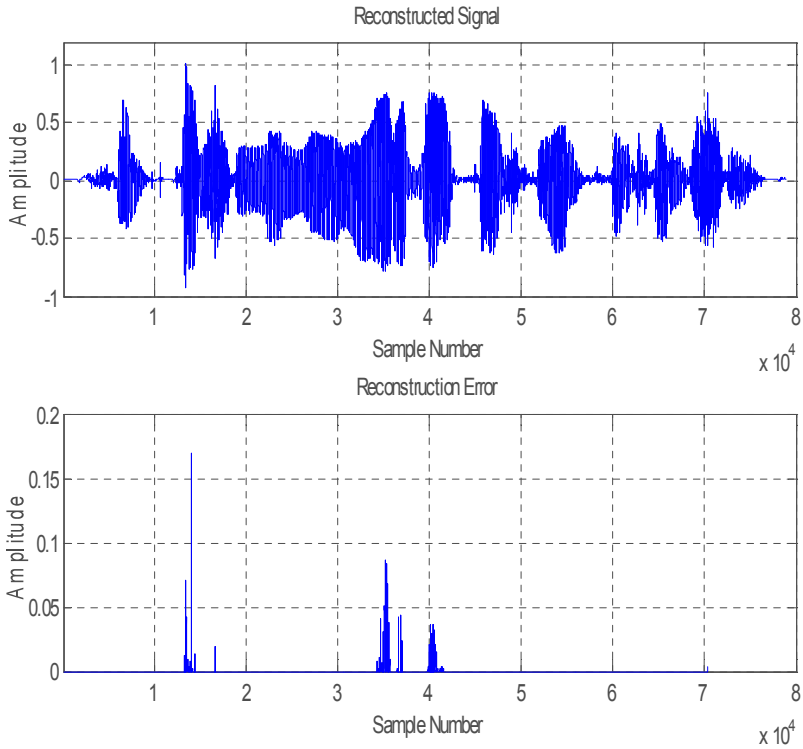


Figure 3-9 Artificially Clipped Natural Speech Reconstruction Error

3.4.3 Clipped Natural Speech

The final test is performed on clipped natural speech. Figure 3-10 and Figure 3-11 show the clipped and the reconstructed signal. It is impossible in this case to present an error plot due to the absence of the original unclipped signal. The only comment that

can be made about the above plots is that the reconstructed signal looks like an unclipped signal. Since there is no reference to objectively evaluate the performance of the algorithm, a subjective criterion is used for judging the quality of the restoration. The speech samples (clipped and restored) were presented to several listeners and they were asked to evaluate the quality of the message by giving a grade between zero and five (zero meaning completely unintelligible and five meaning very clear). The result is a great improvement in intelligibility. The clipped signal was given an average grade of about two while the restored signal received a grade that varied between four and five.

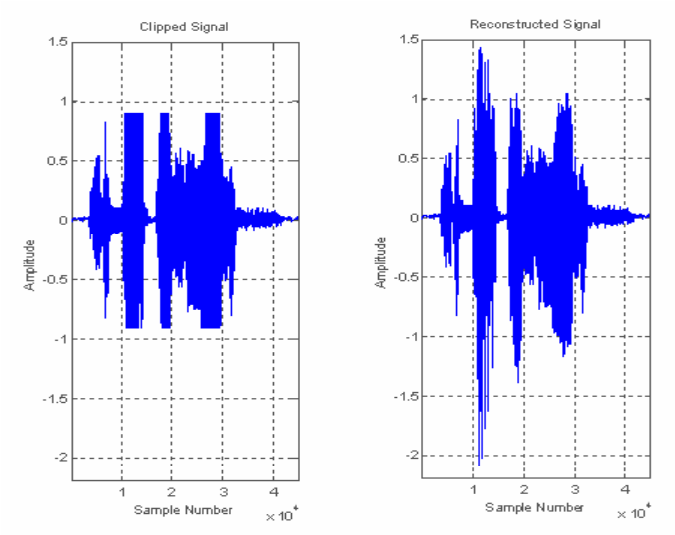


Figure 3-10 Clipped Natural Speech Backward Reconstruction using Least Square Estimation

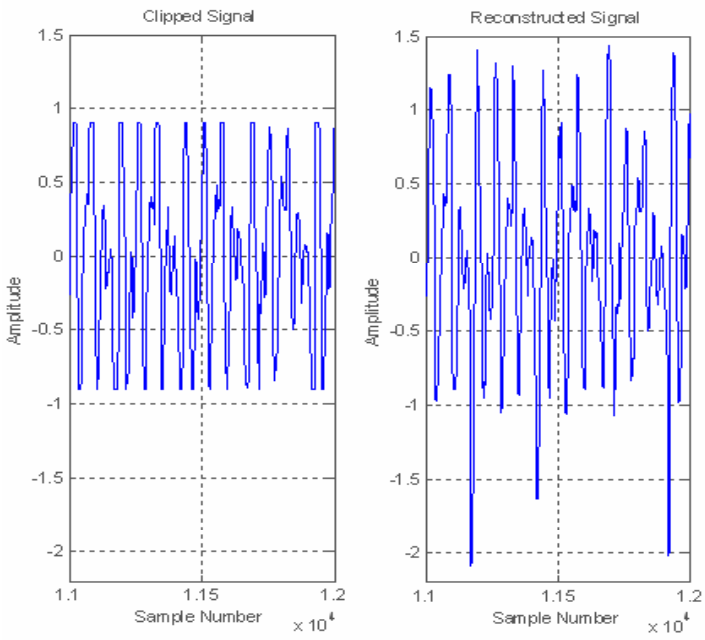


Figure 3-11 Zoomed segment of the reconstruction process

3.4.4 Discussion

It is quite hard to provide a figure of merit for the method other than the plot of the error signal between the unclipped and the reconstructed speech signal. We can observe from the plots in Figure 3- 5 and Figure 3-9 that two parameters determine the quality of the reconstruction: the amplitude and the duration of the error spikes. We can resume both parameters in the following quality factor:

$$\Gamma = \frac{1}{N_c}\sum_{k=1}^{N_c}\left|e(k)\right| \tag{3.16}$$

where $e(k)$ is the error signal between the unclipped and the reconstructed speech signal and N_c is the number of clipped samples. Figure 3- 12 shows the quality factor for forward and backward reconstruction as a function of the number of clipped samples per pitch period for synthetic speech. For clipped natural speech, it of course impossible to provide such data. For artificially clipped natural speech, the quality factor curve using backward reconstruction is so close to the one for forward reconstruction for synthetic speech that it is impossible to provide a separate plot. From the different plots (Figure 3- 5, Figure 3-9 and Figure 3- 12), we can conclude that the estimation of parameters using least square followed by a backward reconstruction offers the best results in term of accuracy.

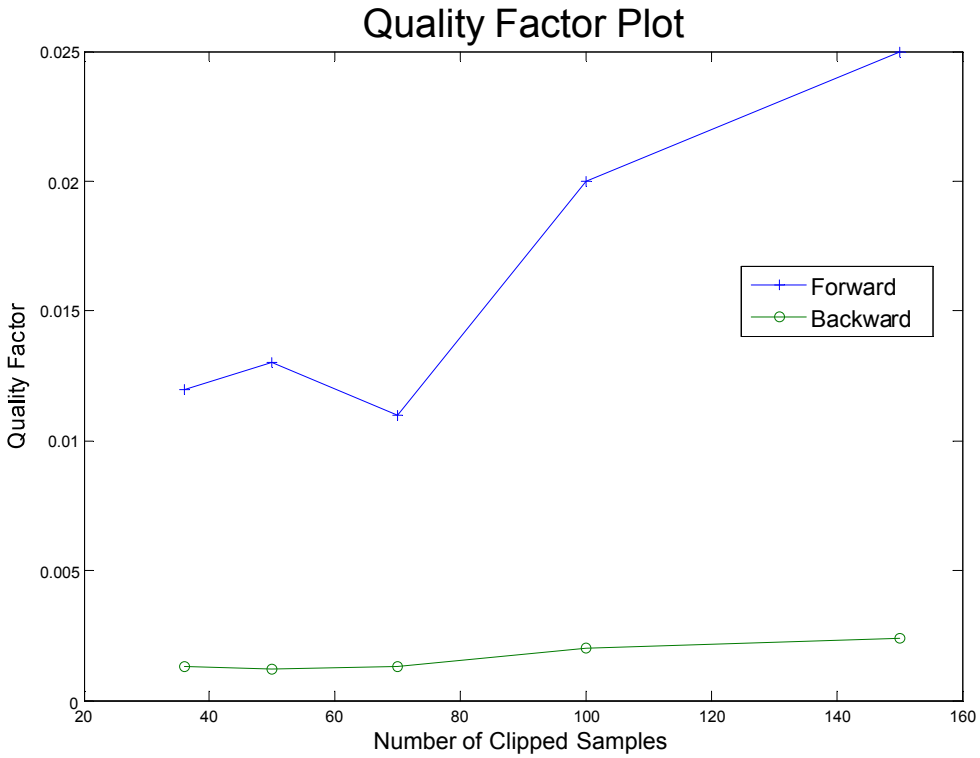


Figure 3- 12 Plot of Quality Factor vs Number of Clipped Samples for Synthetic Speech

Finally, we settled for a least square estimation followed by backward reconstruction as a front end in the pre-processing part of our speech recognition software. The next step in signal processing is noise removal part which will be described in the next chapter.

Chapter 4

Speech Denoising using Wiener Filters

We have seen in chapter 3 that the signal acquired under normal condition using a standard PC sound card and a general purpose microphone contains not only the needed information but also ambient noise. The only way to avoid this ambient noise is to use a sound anechoic chamber to record the speech signal. In this chapter, we are going to analyse ways to eliminate (or at least to reduce) noise.

Noise can dramatically degrade a speech recognition rate. The recognition rate of a single letter recognizer will drop from 87% for clean speech to about 75% in relatively good noise conditions with an average signal to noise ratio (SNR) of 15 dB to a very small value at $\mathrm{SNR} \approx 0$ dB [42]. In this case, noise reduction software is mandatory. Noise reduction software can also be used to increase the comfort or the intelligibility of a phone conversation.

Denoising methods depend on the number of microphones used to acquire the signal. If we have a large number of microphones (more than two), we can use classical adaptive array processing to zero the signal in the noise direction and enhance the signal in the speaker's direction [91]. With two microphones, we can use adaptive noise cancellation [92]. In this case, one microphone should pick mainly surrounding noise while the other will acquire the signal plus the noise. In the case that interests us, we assume that we have only one microphone available. In this case, we have to rely on assumptions about the speech and the noise signals. We are going to use statistical mean square estimation theory in order to "clean out" the speech signal. A suitable mean square method is Wiener filter theory [100].

4.1 Wiener Filter Theory

Wiener filtering is basically a method of signal estimation [68, 84, 92]. It is based on minimizing a mean square error between a signal to be estimated and an estimate. The signal to be estimated is the clean speech signal $x(n)$ and the estimate is the output of the filter $\hat{x}(n)$. The estimate is obtained by linear filtering a noisy speech signal $y(n)$ as shown in Figure 4-1.

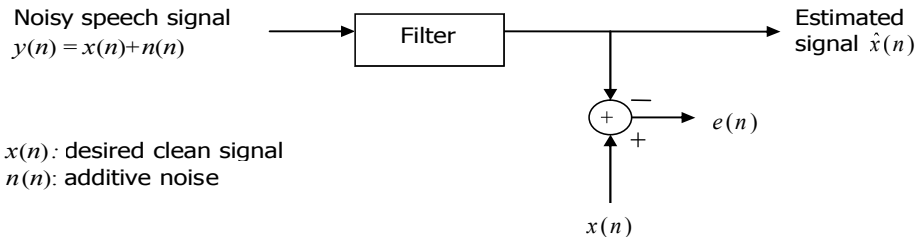


Figure 4-1 Time Domain Structure of the Wiener Filter

The difference $e(n)$ between the desired signal $x(n)$ and the filter output $\hat{x}(n)$ is the error signal. The Wiener filter is the optimum filter that minimizes the mean of the square of $e(n)$. This is known as the "minimum mean square estimation": MMSE. The mean square error is given by:

$$\begin{aligned} P &= E\left[\left(e(n)\right)^2\right] \\ &= E\left[\left(x(n)-\hat{x}(n)\right)^2\right] \end{aligned} \tag{4.1}$$

where $E[\ .\ ]$ indicates statistical expectation.

According to the block diagram shown in Figure 4-1, we can write the following relation between $y(n)$ and $\hat{x}(n)$:

$$\hat{x}(n) = \sum_{k \in I} h(k)\, y(n-k) \tag{4.2}$$

where the limits of the summation depend on the nature of the filter $h(n)$.

In order to solve the problem, we have to make a certain number of assumptions. The first one is that the different signals are stationary. We have seen in chapter one that if we observe the speech signal over a time interval that does not exceed 20 ms, we can safely assume stationarity. The second assumption is that the estimator (the filter) is time invariant. We are going to drop this second requirement in the second part of the chapter. Applying the principle of orthogonality [68], we obtain the following relation:

$$r_{yx}(n) = \sum_{k \in I} h_{opt}(k)\, r_{yy}(n-k) \qquad ; n \in I \tag{4.3}$$

Equation (4.3) is better known as the "Wiener-Hopf" equation. r_{yy} and r_{yx} are respectively the noisy signal autocorrelation function and the noisy and clean signal cross-correlation function. h_{opt} is the impulse response of the Wiener filter, i.e. the

optimum filter $h(n)$. Depending on the limits of the above summation, we have the following three different cases:

- Causal finite impulse response (FIR) filter if $I = \{0, 1, \ldots, N - 1\}$.

- Non causal infinite impulse response (IIR) filter if $I =]-\infty, +\infty[$

- Causal IIR filter if $I =]0, +\infty[$

When we use the optimum filter, the mean square error becomes:

$$P_0 = r_{xx}(0) - \sum_{k \in I} h_{opt}(k) r_{yx}(k) \tag{4.4}$$

Equation (4.4) can be easily computed for FIR filters.

4.1.1 Non Causal IIR Wiener Filters

In the non causal case, the Wiener-Hopf equation (4.3) becomes:

$$r_{yx}(n) = \sum_{k=-\infty}^{+\infty} h_{nc}(k)\, r_{yy}(n-k) \tag{4.5}$$

h_{nc} stands for the non causal impulse response. Equation (4.5) shows that in this case, the cross correlation r_{yx} is computed by the convolution of the impulse response h_{nc} with the autocorrelation r_{yy}.

$$r_{yx}(n) = h_{nc}(n) * r_{yy}(n) \tag{4.6}$$

Taking the z-transform of the above relation, we obtain the transfer function of the non causal IIR Wiener filter:

$$H_{nc}(z) = \frac{R_{yx}(z)}{R_{yy}(z)} \tag{4.7}$$

Assuming that the signal $x(n)$ and the noise $n(n)$ are uncorrelated processes and that the noise is zero mean, each one of the above power spectral densities (PSD) can be expressed as:

$$R_{yy}(z) = R_{xx}(z) + R_{nn}(z) \tag{4.8}$$

$$R_{yx}(z) = R_{xx}(z) \tag{4.9}$$

If we want to implement the filter, we have to estimate the above the PSD's $R_{xx}(z)$ and $R_{nn}(z)$. We will use autoregressive (AR) modeling. Finally, the transfer function of the non causal filter is:

$$H_{nc}(z) = \frac{R_{xx}(z)}{R_{xx}(z) + R_{nn}(z)} \tag{4.10}$$

Using Wold representation [68], each one of the PSD's $R_{yy}(z)$ and $R_{nn}(z)$ can be factored as:

$$R_{yy}(z) = \sigma_w^2 H_y(z) H_y(z^{-1}) \tag{4.11}$$

$$R_{nn}(z) = \sigma_w^2 H_n(z) H_n(z^{-1}) \tag{4.12}$$

Using only the observed signal $y(n)$ and the noise signal $n(n)$, we can express the filter transfer function as:

$$H_{nc}(z) = \frac{R_{xx}(z)}{R_{yy}(z)} = \frac{H_y(z)H_y(z^{-1}) - H_n(z)H_n(z^{-1})}{H_y(z)H_y(z^{-1})} \tag{4.13}$$

and finally:

$$H_{nc}(z) = \frac{1}{H_y(z)} \left[\frac{R_{xx}(z)}{\sigma_w^2 H_y(z^{-1})} \right] \tag{4.14}$$

Therefore, we can say that the filter which estimates $x(n)$ from $y(n)$ is obtained by cascading the whitening filter $\frac{1}{H_y(z)}$ with the optimum filter that estimates $x(n)$ from the innovation process $w(n)$ as shown in the following figure [68]:

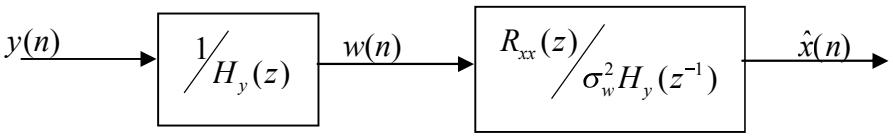


Figure 4-2 Block Diagram of the Non-Causal IIR Wiener Filter

4.1.2 Causal IIR Wiener Filter

In the causal IIR case, the Wiener-Hopf equation becomes:

$$r_{yx}(n) = \sum_{k=0}^{+\infty} h_c(k) r_{yy}(n-k) \qquad ; \quad n \geq 0 \tag{4.15}$$

In this case, we cannot use simple z-transform of a convolution because of the causality constraint. We have to resort to a spectral factorization method that was developed by Wiener [100, 68] for rational PSD's. The causal IIR Wiener filter will be

obtained by cascading the whitening filter $\frac{1}{H_y(z)}$ with a causal filter that estimates $x(n)$ from the innovation process $w(n)$. The causal transfer function is:

$$H_c(z) = \frac{1}{H_y(z)} \left[\frac{R_{xx}(z)}{\sigma_w^2 H_y(z^{-1})} \right]_+ \tag{4.16}$$

$[.]_+$ denotes the causal part of the expression inside the brackets.

4.1.3 Causal FIR Wiener Filter

Finally, in the causal FIR filter of length N, the Wiener-Hopf equation becomes:

$$r_{yx}(n) = \sum_{k=0}^{N-1} h_F(k) r_{yy}(n-k) \qquad ;\ 0 \le n \le N-1 \tag{4.17}$$

Equation 4.17 represents N equations with N unknowns. It can be expressed in the following matrix form:

$$\begin{pmatrix} r_{xx}(0) \\ r_{xx}(1) \\ r_{xx}(2) \\ \vdots \\ r_{xx}(N-1) \end{pmatrix} = \begin{pmatrix} r_{yy}(0) & r_{yy}(1) & r_{yy}(2) & \cdots & r_{yy}(N-1) \\ r_{yy}(1) & r_{yy}(0) & r_{yy}(1) & \cdots & r_{yy}(N-2) \\ r_{yy}(2) & r_{yy}(1) & r_{yy}(0) & \cdots & r_{yy}(N-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{yy}(N-1) & r_{yy}(N-2) & r_{yy}(N-3) & \cdots & r_{yy}(0) \end{pmatrix} \begin{pmatrix} h_F(0) \\ h_F(1) \\ h_F(2) \\ \vdots \\ h_F(N-1) \end{pmatrix} \tag{4.18}$$

or:

$$\mathbf{r}_{xx} = \mathbf{R}_{yy} \mathbf{h}_F \tag{4.19}$$

In this particular case, the mean square error can be evaluated as:

$$P_{0F} = r_{xx}(0) - \sum_{m=0}^{N-1} h_F(m) r_{yx}(m) \tag{4.20}$$

4.2 The Autoregressive Model

When we want to solve the Wiener estimation using IIR filters, we have to estimate the power spectra $R_{xx}(z)$ and $R_{nn}(z)$. The Wiener factorization method can be applied when these power spectra are rational functions. According to Wold representation, a wide sense stationary random process $x(n)$ may be represented as the output of a causal and causally invertible (minimum phase) linear time invariant system excited by a white noise $w(n)$ as shown in Figure 4-3. The white noise $w(n)$ is called the innovation process associated with the process $x(n)$.

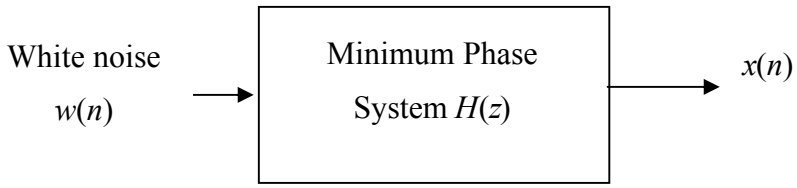


Figure 4-3 Generation of the Process *x*(*n*)

If the variance of the innovation process is σ_w^2, we can express the PSD of the process $x(n)$ as [68]:

$$R_{xx}(z) = \sigma_w^2 H(z) H(z^{-1}) \tag{4.21}$$

We say that the process $x(n)$ is modeled using an autoregressive (AR) model if the transfer function $H(z)$ is all pole. At that time, we can express it as:

$$H(z) = \frac{\alpha}{1 - \sum_{k=1}^{P} a_k z^{-k}} \tag{4.22}$$

α is the gain of the model, P is the order and the a_k 's are the model coefficients that determine the poles of the model. We can compute these coefficients using the Yule-Walker equations [68, 84]:

$$\begin{pmatrix} r_{xx}(0) & -r_{xx}(1) & \cdots & -r_{xx}(P) \\ r_{xx}(1) & -r_{xx}(0) & \cdots & -r_{xx}(P-1) \\ \vdots & \vdots & \vdots & \vdots \\ r_{xx}(P) & -r_{xx}(P-1) & \cdots & -r_{xx}(0) \end{pmatrix} \begin{pmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{pmatrix} = \begin{pmatrix} \alpha^2 \sigma_w^2 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{4.23}$$

The solution is provided by:

$$\begin{pmatrix} a_1 \\ \vdots \\ a_p \end{pmatrix} = \begin{pmatrix} r_{xx}(0) & \cdots & r_{xx}(P-1) \\ \vdots & \vdots & \vdots \\ r_{xx}(P-1) & \cdots & r_{xx}(0) \end{pmatrix}^{-1} \begin{pmatrix} r_{xx}(1) \\ \vdots \\ r_{xx}(P) \end{pmatrix} \tag{4.24}$$

and

$$\alpha = \sqrt{\frac{1}{\sigma_w^2} \begin{pmatrix} r_{xx}(0) & -r_{xx}(1) & \cdots & -r_{xx}(P) \end{pmatrix} \begin{pmatrix} 1 & a_1 & \cdots & a_P \end{pmatrix}^{\mathrm{T}}} \tag{4.25}$$

The autocorrelation $r_{xx}(k)$ is computed using the usual formula [66]:

$$r_{xx}(k) = \frac{1}{L-|k|} \sum_{n=0}^{L-k-1} x(n+k)x(n) \quad ; \; k \geq 0 \tag{4.26}$$

L is the size of the observation window.

4.3 Time Invariant Wiener Filtering

The first experiments concern a short word pronounced a male speaker (the English word "one"). The ambient noise was produced by a radio tuned at an arbitrary frequency in such a way that no station was picked by the antenna. The data was recorded with a sampling frequency of 44.1 kHz and a quantization level of 16 bits. The data was stored in a file. This allows non causal signal processing.

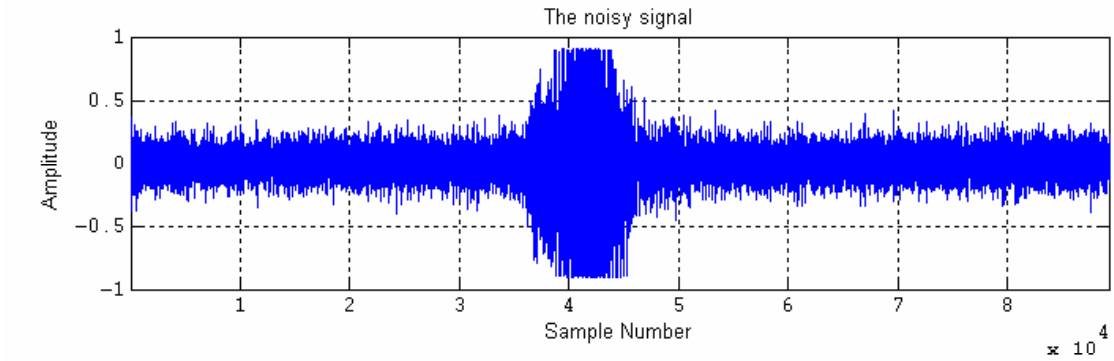


Figure 4-4 Recorded Noisy Speech Signal

4.3.1 FIR Time Invariant Wiener Filter

We are going to use FIR filters to clean the signal shown in Figure 4-4. We use three different orders: $N = 8$, $N = 64$ and $N = 512$. Equation 4.18 is used to compute the filter impulse response h_F. Then, the signal is filtered by computing the convolution of the signal with the impulse response.

Figure 4-5 shows the filtered signal for $N = 8$. We can notice a significant noise reduction in the filtered signal. The signal shown in Figure 4-5 appears much clearer than the original one in a hearing test.

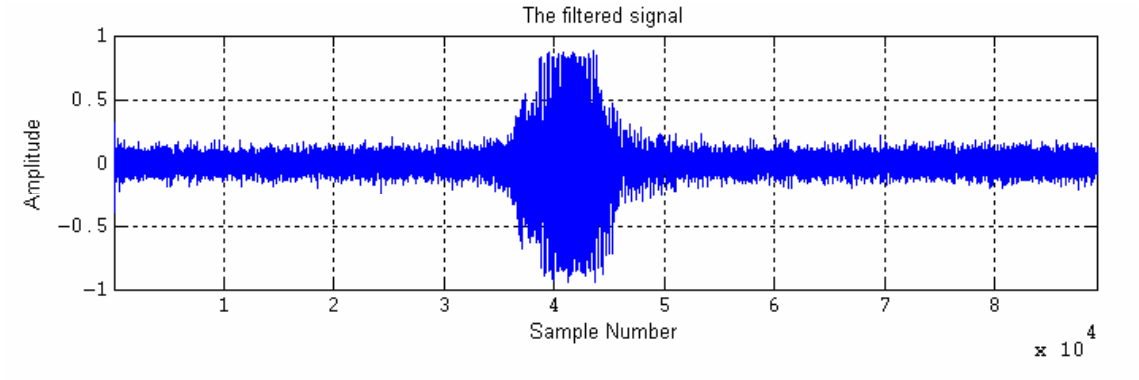


Figure 4-5 Filtered Speech Signal for $N = 8$

Repeating the same procedure for $N = 64$ and $N = 512$ yields the following results.

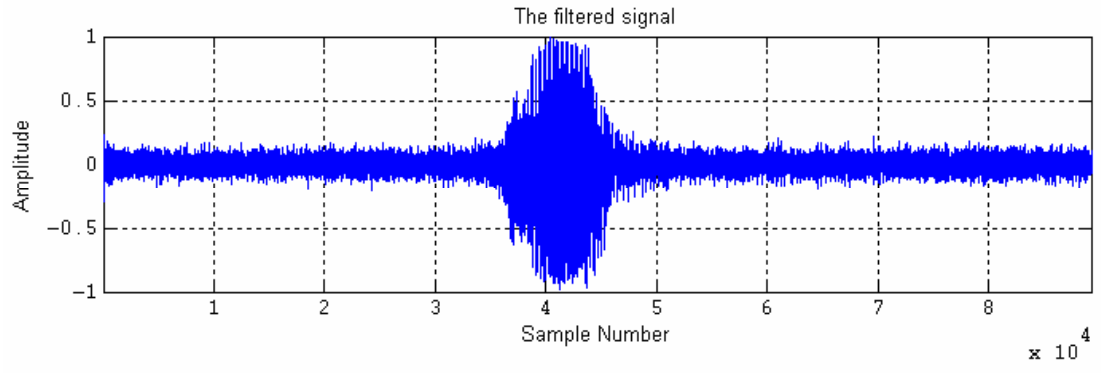


Figure 4-6 Filtered Speech Signal for $N = 64$

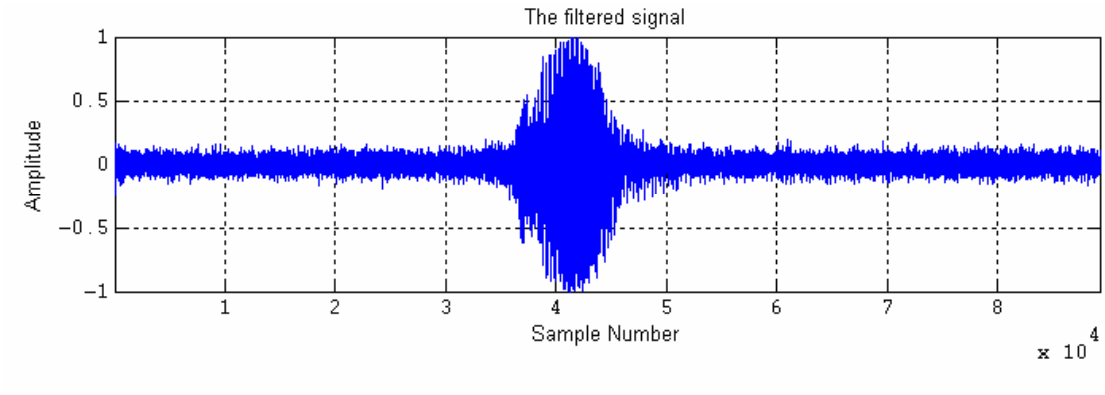


Figure 4-7 Filtered Speech Signal for $N = 512$

Comparing Figure 4-5, Figure 4-6 and Figure 4-7, we remark that as N increases, the noise level is reduced. However, we also remark that there is no much reduction between the last two figures. This is illustrated by the following plot (Figure 4-8) of the

mean square error P_{0F} versus the filter order N. The value of the mean square error is computed using equation 4.20. It is clear that a value of N larger than 32 will not reduce significantly the mean square error.

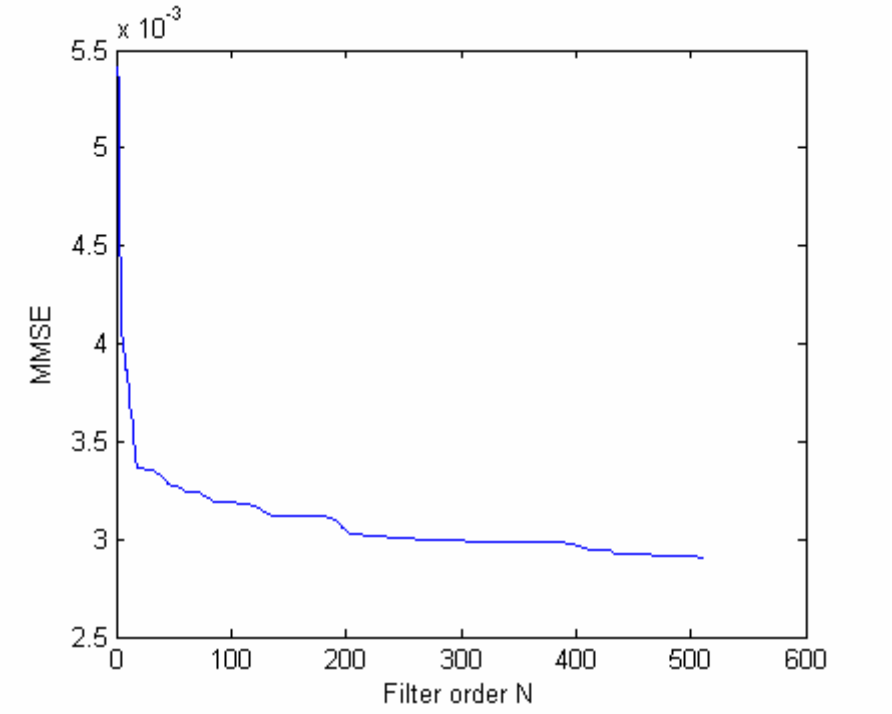


Figure 4-8 Mean Square Error vs Filter Order

We can also observe that the signal is more and more distorted as N increases. From the distortion point of view, we can conclude that the smaller order is better.

4.3.2 IIR Causal Wiener Filter

In this section, the noisy speech and the ambient noise are going to be modelled by autoregressive models of order one and two. From the obtained transfer functions, we realize the filters by implementing difference equations.

4.3.4.a First Order Model

Let:

$$H_y(z) = \frac{\alpha}{1 - a_1 z^{-1}} \tag{4.27}$$

$$H_n(z) = \frac{\beta}{1 - b_1 z^{-1}} \tag{4.28}$$

The model parametersα, a_1, β and b_1 are computed using the Yule-Walker equations. Assuming a unity variance innovation process, the power spectra for the noisy signal $y(n)$ and the noise $n(n)$ are given by:

$$R_{yy}(z) = \frac{\alpha^2}{(1-a_1 z^{-1})(1-a_1 z)} \tag{4.29}$$

$$R_{nn}(z) = \frac{\beta^2}{(1-b_1 z^{-1})(1-b_1 z)} \tag{4.30}$$

Equation (4.16) is developed using the above relations to give:

$$\left[\frac{R_{xx}(z)}{\sigma_w^2 H_y(z^{-1})}\right] = \frac{A_1 z^{-1} + B_1 + C_1 z}{(1-a_1 z^{-1})(1-b_1 z^{-1})(1-b_1 z)}$$

$$= \frac{r_1}{1-a_1 z^{-1}} + \frac{r_2}{1-b_1 z^{-1}} + \frac{r_3 z}{1-b_1 z} \tag{4.31}$$

where A_1, B_1, C_1, r_1, r_2 and r_3 are real valued constants. Taking the causal part of the above equation yields:

$$\left[\frac{R_{xx}(z)}{\sigma_w^2 H_y(z^{-1})}\right]_+ = \frac{r_1}{1-a_1 z^{-1}} + \frac{r_2}{1-b_1 z^{-1}} \tag{4.32}$$

So, finally, using equation (4.16), we obtain the following transfer function:

$$H_c(z) = \frac{A z^{-1} + B}{1-b_1 z^{-1}} = \frac{\hat{x}(z)}{y(z)} \tag{4.33}$$

Therefore, the difference equation implementing the IIR causal Wiener filter is:

$$\hat{x}(n) = b_1 \hat{x}(n-1) + B\, y(n) + A\, y(n-1) \tag{4.34}$$

Using the Yule-Walker equations, we evaluate the first order model parameter from the signal and the noise shown in Figure 4-4. These coefficients are recorded in Table 4-1.

	Model Coefficients	Gain
Corrupted speech signal	$a_1 = 0.9591$	$\alpha = 0.0459$
Ambient noise	$b_1 = 0.9005$	$\beta = 0.0380$

Table 4-1 First Order Model Parameters

We next calculate the filter coefficients to implement the digital filter. They are given in Table 4-2.

Filter Coefficients
A= -0.4260
B= 0.5053

Table 4-2 Filter Coefficients

Applying this filter to the noisy speech signal produces the following result:

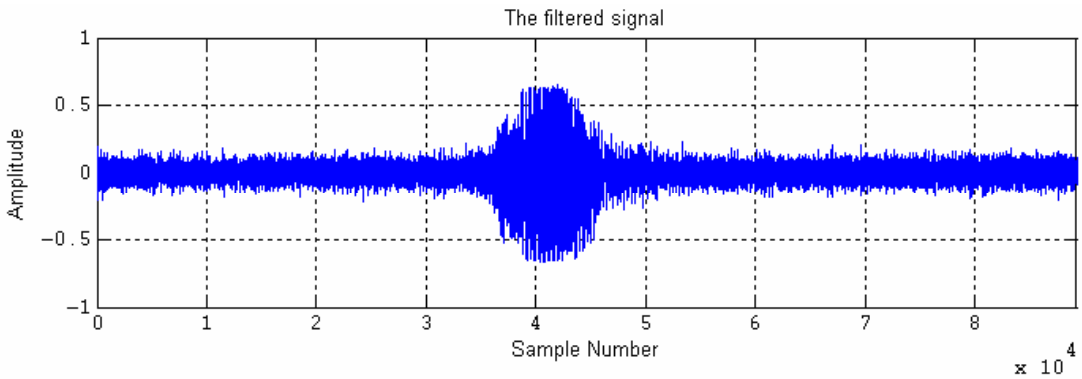


Figure 4-9 Filtered Speech Signal using Causal First Order Filter

4.3.4.b Second Order Model

Let:

$$H_y(z) = \frac{\alpha}{(1 - a_1 z^{-1} - a_2 z^{-2})}$$

$$= \frac{\alpha}{(1 - p_1 z^{-1})(1 - p_2 z^{-1})} \tag{4.35}$$

$$H_n(z) = \frac{\beta}{(1 - b_1 z^{-1} - b_2 z^{-2})}$$

$$= \frac{\beta}{(1 - q_1 z^{-1})(1 - q_2 z^{-1})} \tag{4.36}$$

where p_1, p_2 and q_1, q_2 are the poles of $H_y(z)$ and $H_n(z)$ respectively. $\alpha, a_1, a_2, \beta, b_1$ and b_2 are given by the Yule-Walker equations. Assuming a unity variance

innovation process, the power spectra for the noisy signal $y(n)$ and the noise $n(n)$ are given by:

$$R_{yy}(z) = \frac{\alpha^2}{(1-p_1 z^{-1})(1-p_1 z)(1-p_2 z^{-1})(1-p_2 z)} \tag{4.37}$$

$$R_{nn}(z) = \frac{\beta^2}{(1-q_1 z^{-1})(1-q_1 z)(1-q_2 z^{-1})(1-q_2 z)} \tag{4.38}$$

Therefore, following the same work done for the first order model, we obtain the transfer function:

$$\begin{aligned} H_c(z) &= \frac{A z^{-3} + B z^{-2} + C z^{-1} + D}{(1-q_1 z^{-1})(1-q_2 z^{-1})} \\ &= \frac{A z^{-3} + B z^{-2} + C z^{-1} + D}{1-b_1 z^{-1} - b_2 z^{-2}} = \frac{\hat{x}(z)}{y(z)} \end{aligned} \tag{4.39}$$

All the constants of the above transfer function are real numbers. So, the filter will be implemented using the following difference equation:

$$\hat{x}(n) = b_1\hat{x}(n-1) + b_2\hat{x}(n-2) + D\,y(n) + C\,y(n-1) + B\,y(n-2) + A\,y(n-3) \tag{4.40}$$

Using the Yule-Walker equations, we evaluate the first order model parameter from the signal and the noise shown in Figure 4-4. These coefficients are recorded in Table 4-3.

	Model Coefficients	Gain
Corrupted speech signal	$a_1 = 1.7525$ $a_2 = -0.8274$	$\alpha = 0.0258$
Ambient noise	$b_1 = 1.6607$ $b_2 = -0.8443$	$\beta = 0.0204$

Table 4-3 Second Order Model Parameters

We next calculate the filter coefficients to implement the digital filter. They are given in Table 4-4. When applied to the noisy speech signal, the filter produces the results displayed in Figure 4-10.

Filter coefficients
A= -0.8597
B= 3.0397
C= -3.4930
D= 1.4526

Table 4-4 Filter Coefficients

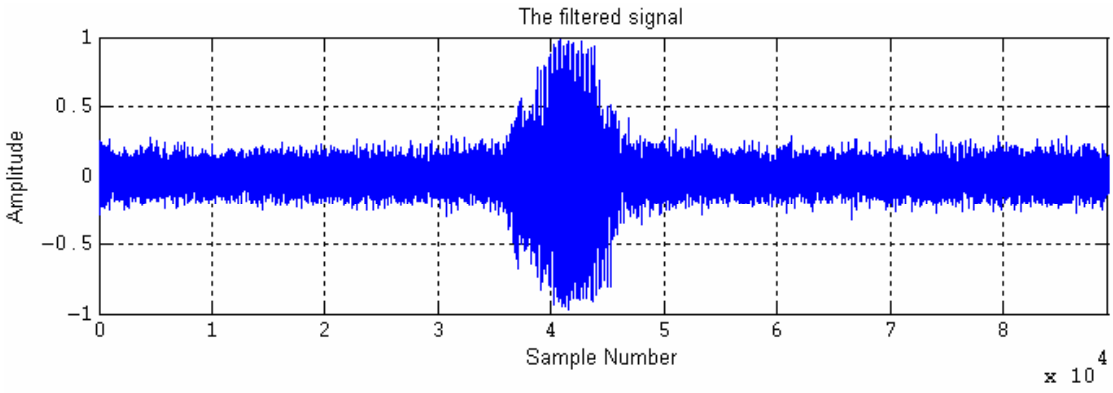


Figure 4-10 Filtered Speech Signal using Causal Second Order Filter

We remark that the second order filter passes more noise than the first order one. However, it produces less distortion. This behaviour is typical of Wiener filters. The output is a compromise between good noise elimination and good signal preservation.

4.3.3 IIR Non Causal Wiener Filter

The power spectra models (first and second order) have been computed in the previous section, we just have to use equation (4.13) to obtain the transfer function of the non causal IIR filter of order one and two.

4.3.3.a First Order Model

The power spectra are given by equations (4.29) and (4.30). We deduce that the first order non causal IIR filter has the following transfer function:

$$H_{nc}(z) = \frac{A\,z^{-2} + B\,z^{-1} + C}{(1 - b_1\,z^{-1})(1 - b_1^{-1} z^{-1})} = K + \frac{r_1}{1 - b_1\,z^{-1}} + \frac{r_2}{1 - b_1^{-1} z^{-1}} \tag{4.41}$$

where A, B, C, K, r_1 and r_2 are real constants.

The first and second term of the partial fraction expansion of $H_{nc}(z)$ describe stable causal systems. They can be implemented using the following difference equations:

$$\hat{x}_1(n) = K\,y(n) \tag{4.42}$$

$$\hat{x}_2(n) = b_1\,\hat{x}_2(n-1) + r_1\,y(n) \tag{4.43}$$

The third term describes a stable anti-causal system. It can be implemented by processing the data backward (from the end of the file to the beginning) through the entire length of the noisy signal using the following difference equation:

$$\hat{x}_3(n-1) = b_1\,(\hat{x}_3(n) - r_2\,y(n)) \tag{4.44}$$

The estimated signal $\hat{x}(n)$ is the sum of $\hat{x}_1(n)$, $\hat{x}_2(n)$ and $\hat{x}_3(n)$.

For the implementation, we use the model parameters collected in Table 4-1. The filter coefficients are given in Table 4-5.

Filter coefficients
A= 0.2693
B= -0.5484
C= 0.2693

Table 4-5 Filter Coefficients

Figure 4-11 displays a pole and zero plot of the transfer function in the complex plane. The transfer function of the filter shows a pair of real poles, one inside and one outside the unit circle. The pole inside the unit circle is the one of the stable causal part of the transfer function whereas the pole outside the unit circle is the one of the stable anti-causal part.

The noisy speech signal is filtered using the three difference equations (4.42), (4.43), (4.44). It is displayed in Figure 4-12.

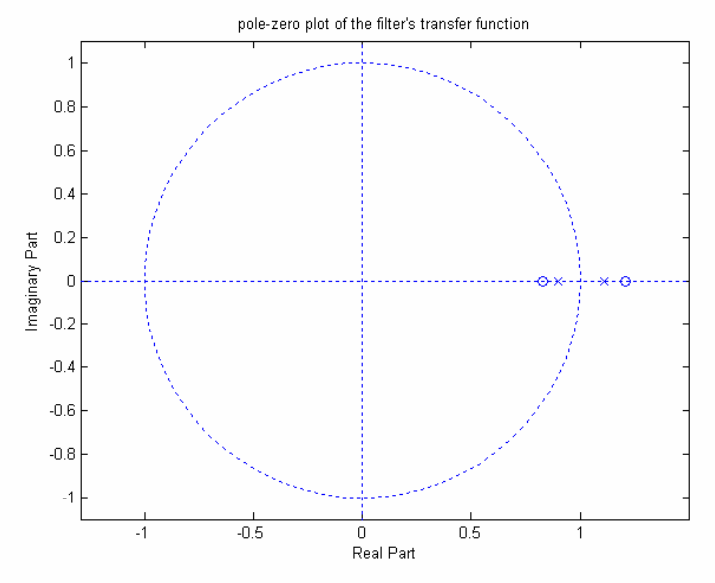


Figure 4-11 First Order Pole and Zero Plot

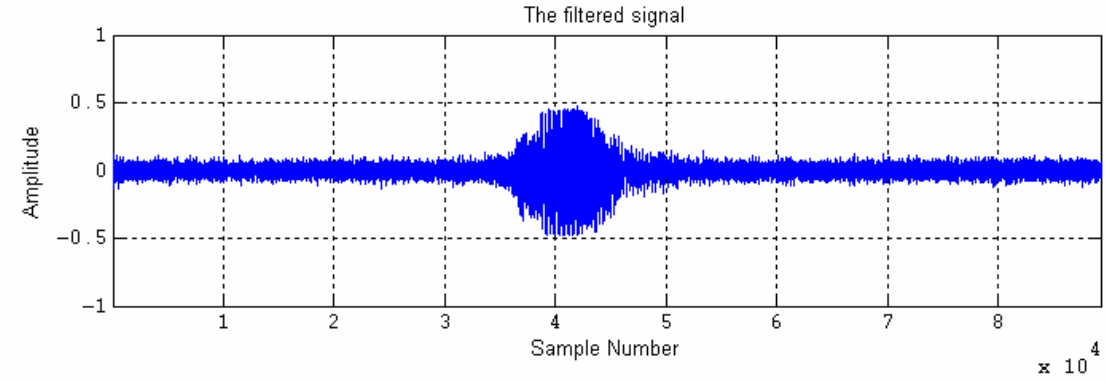


Figure 4-12 Filtered Speech Signal using Non-Causal First Order Filter

4.3.3.b Second Order Model

The power spectra are given by equations (4.37) and (4.38). We deduce that the second order non causal IIR filter has the following transfer function:

$$H_{nc}(z) = \frac{A\,z^{-4} + B\,z^{-3} + C\,z^{-2} + D\,z^{-1} + E}{(1-q_1\,z^{-1})(1-q_1^{-1}z^{-1})(1-q_2\,z^{-1})(1-q_2^{-1}z^{-1})}$$
$$= K + \frac{r_1}{1-q_1\,z^{-1}} + \frac{r_2}{1-q_2\,z^{-1}} + \frac{r_3}{1-q_1^{-1}z^{-1}} + \frac{r_4}{1-q_2^{-1}z^{-1}} \qquad (4.45)$$

The residues r_1, r_2, r_3 and r_4 can be either real or complex (conjugate) numbers while K must be a real constant. Grouping the second term with the third one and the

fourth term with the fifth one in equation (4.45) yields the following three difference equations:

$$\hat{x}_1(n) = K\ y(n) \tag{4.46}$$

$$\hat{x}_2(n) = (q_1 + q_2)\hat{x}_2(n-1) - q_1 q_2 \hat{x}_2(n-2) + (r_1 + r_2)y(n) - (r_1 q_2 + r_2 q_1)y(n-1) \tag{4.47}$$

$$\hat{x}_3(n-2) = (q_1 q_2)((q_1^{-1} + q_2^{-1})\hat{x}_3(n-1) - \hat{x}_3(n) + (r_3 + r_4)y(n) - (r_3 q_2^{-1} + r_4 q_1^{-1})y(n-1)) \tag{4.48}$$

Since each pair of (r_1, r_2), (r_3, r_4) and (q_1, q_2) is complex conjugate, the last three difference equations can be implemented directly where equations (4.46) and (4.47) describe stable causal systems, equation (4.48) describes a stable anti-causal system. It will be implemented using backward processing as for the first order system.

The sum of $\hat{x}_1(n)$, $\hat{x}_2(n)$ and $\hat{x}_3(n)$ is the estimated signal $\hat{x}(n)$.

For the implementation, we use the model parameters collected in Table 4-3. The filter coefficients are given in Table 4-6.

Filter coefficients
A= 0.3879
B= -1.2585
C= 1.7769
D= -1.2585
E= 0.3879

Table 4-6 Filter Coefficients

Figure 4-13 is a pole and zero plot of the transfer function.

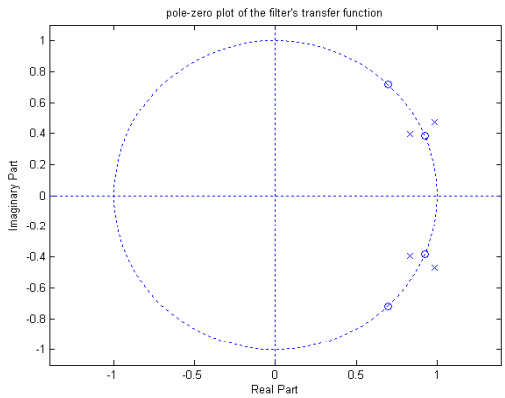


Figure 4-13 Second Order Pole and Zero Plot

The filter presents two pairs of complex conjugate poles. One pair is inside the unit circle while the other is outside the unit circle. The first pair belongs to the stable causal parts of the transfer function. The second pair represents the poles of the stable anti-causal parts. When applied to the noisy speech signal, the filter gives the following result:

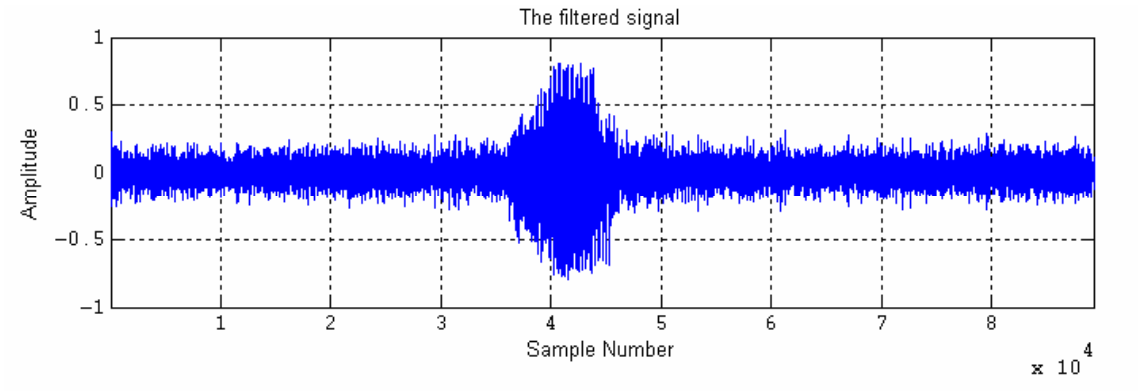


Figure 4-14 Filtered Speech Signal using Non-Causal Second Order Filter

If we compare Figure 4-12 and Figure 4-14, we can say that there is more noise at the output of the second order filter. However, there is much less distortion in the signal filtered by the second order filter. We can also compare the output of the causal and the non causal filters. The non causal filter produces more filtering. However, we always face the same compromise. Either we filter noise and distort the signal or we preserve the signal but keep the noise. We have seen in chapter one that the speech signal is not stationary. A solution for denoising the speech signal is to use time varying Wiener filter [51].

4.4 Time Varying Wiener Filter

A speech segment with duration of more than 20 ms cannot be considered as stationary. This is why it is practically impossible to use efficiently time invariant filters. We are going to use a sliding window to analyse the speech signal and we recomputed the filter parameters every time. It is very hard to implement time varying IIR filters because they have infinite memory (in theory). Therefore, we settle for a causal time varying FIR filter. We use windows of 256 sample-length slid 100 samples at a time from speech waveforms sampled at 44.1 kHz. However, we assume that the noise character remains constant over quite long intervals. A long speech waveform is a succession of silence (noise only) segments and noisy speech segments. We use the noise only segment to evaluate the noise autocorrelation function $r_{nn}(k)$ and the speech segment will be divided into the above mentioned windows in order to evaluate the

noisy speech autocorrelation $r_{yy}(n)$. We need a tool to discriminate between speech and noise. We can use a VAD system based on mean magnitude and zero crossing rate as described in chapter 2. Figure 4-15 shows a signal with two speech regions (between N2 and N3 and between N4 and N5) and three noise-only regions.

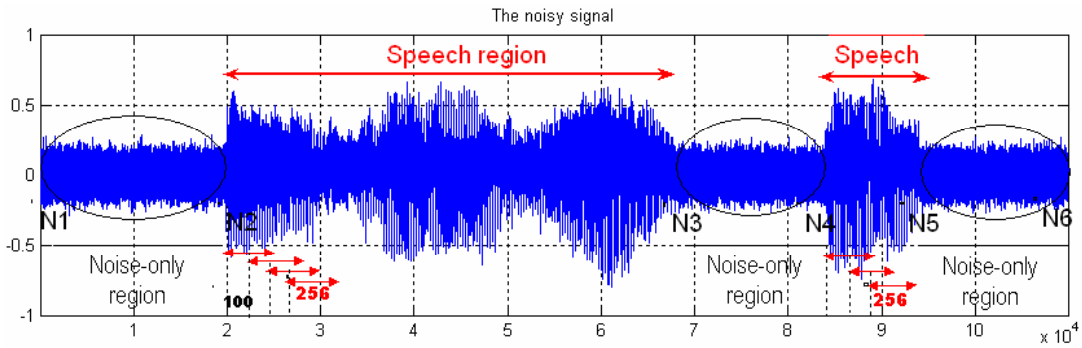


Figure 4-15 Representation of the Speech Windowing

The noise autocorrelation $r_{nn}(k)$ for the above signal will be computed first using the signal from the region between N1 and N2 to compute the filter parameters up to N3. Then, it will be recalculated using the signal between N3 and N4 to compute the filter parameters up to N5, and so on. The noisy speech autocorrelation is recomputed on every 256 sample window as shown above. So, for every speech region, we are going to have L vectors $\mathbf{r}_{yy,k}$, each of dimension N. L is the number of frames, k is the frame index..

Using equation (4.8), we compute the vector $\mathbf{r}_{xx,k}$ as:

$$\mathbf{r}_{xx,k} = \mathbf{r}_{yy,k} - \mathbf{r}_{nn} \tag{4.49}$$

Then, the k^{th} frame impulse response is given by the same relation as equation (4.18), with the major difference being that it has to be recomputed every frame as shown in equation (4.50).

$$\begin{pmatrix} r_{xx,k}(0) \\ r_{xx,k}(1) \\ r_{xx,k}(2) \\ \vdots \\ r_{xx,k}(N-1) \end{pmatrix} = \begin{pmatrix} r_{yy,k}(0) & r_{yy,k}(1) & r_{yy,k}(2) & \cdots & r_{yy,k}(N-1) \\ r_{yy,k}(1) & r_{yy,k}(0) & r_{yy,k}(1) & \cdots & r_{yy,k}(N-2) \\ r_{yy,k}(2) & r_{yy,k}(1) & r_{yy,k}(0) & \cdots & r_{yy,k}(N-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{yy,k}(N-1) & r_{yy,k}(N-2) & r_{yy,k}(N-3) & \cdots & r_{yy,k}(0) \end{pmatrix} \begin{pmatrix} h_{F,k}(0) \\ h_{F,k}(1) \\ h_{F,k}(2) \\ \vdots \\ h_{F,k}(N-1) \end{pmatrix} \tag{4.50}$$

After the computation of the impulse response, the output $\hat{x}(n)$ is computed via convolution of $h_{F,k}(n)$ with the noisy signal $y(n)$. However, since the filter parameters

are dependent only on the speech region, we have to adopt a particular strategy for the noise-only region.

When $y(n)$ consists of only noise, we have two possibilities. We can eliminate the noise by simply setting its value to zero, that is, multiplying it by zero. This method presents many drawbacks. First, the abrupt change will provoke a discontinuity at the boundary between a silence region and a speech region. Next, if the VAG system is not very accurate, we can eliminate information if a speech region is mistakenly included in the silence region. To avoid the above-mentioned drawbacks, we are going to filter the noise by the impulse response computed in the first speech region that immediately follows the concerned silence region.

When $y(n)$ consists of noisy speech, the filter parameters are changed every 100 samples. These parameters are computed using the corresponding window of 256-sample length.

We are going to compare first our time varying filter with the time invariant one of section 4.3.1. The first signal to be filtered is the one represented in Figure 4-4. Figure 4-16 shows the output of a time invariant FIR Wiener filter of order $N = 256$. Figure 4-17 shows the result of using our time varying filter on the same signal. The length of the impulse response is $N = 2$ only. We observe that the second order time varying filter achieves a better noise rejection than the 256-sample long time invariant filter. Furthermore, the time invariant filter distorts the signal while we observe very little signal distortion in the output of the time varying filter.

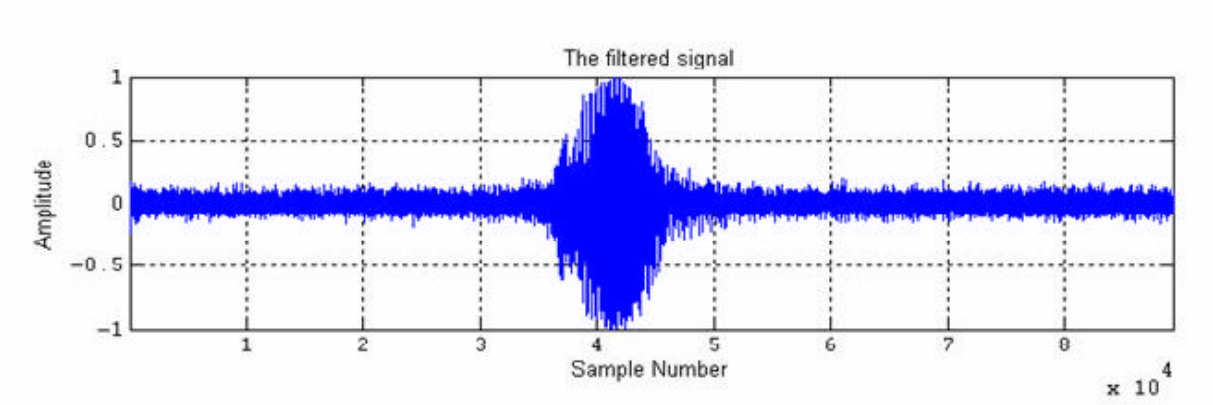


Figure 4-16 Filtered Signal with aTime Invariant Wiener Filter of Order $N = 256$

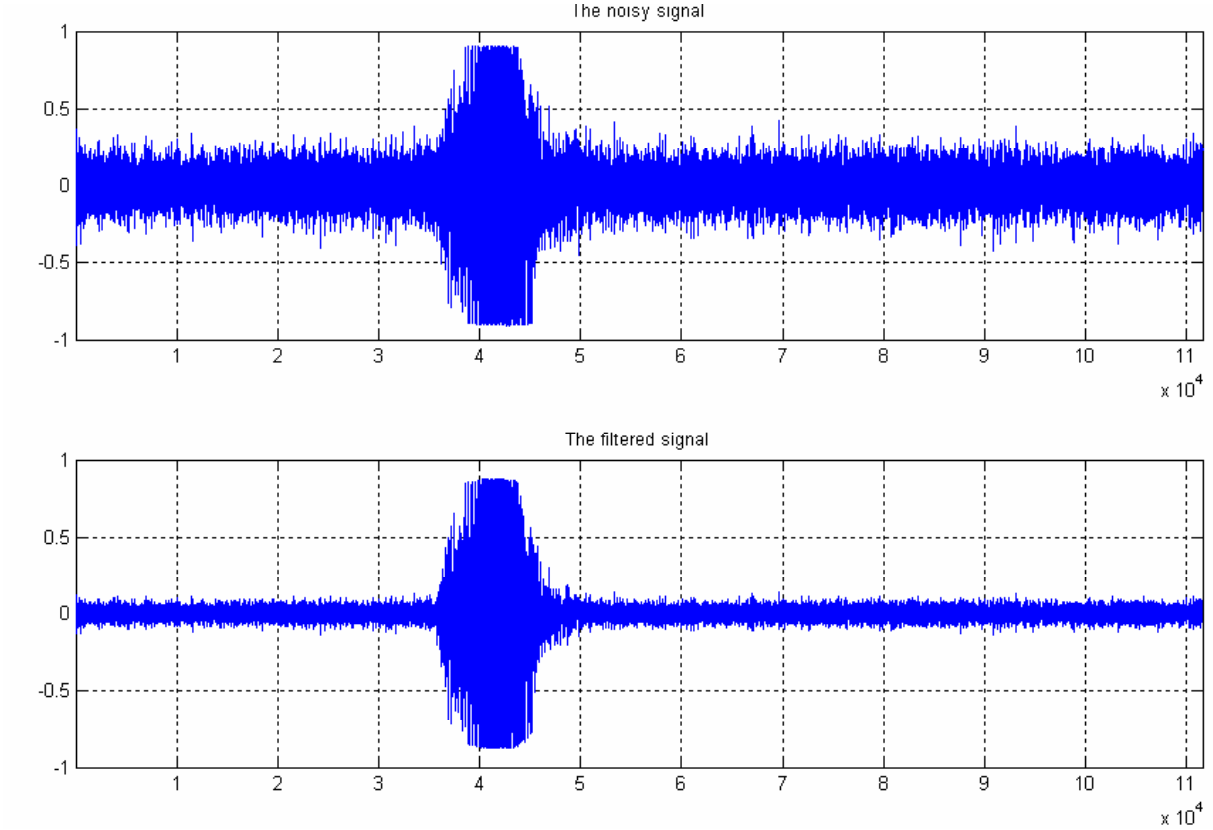


Figure 4-17 Noisy and Filtered Signal with a Time Varying Filter of Order $N = 2$

We then test the algorithm with longer test sentences. The next signal consists of the sentence "they study" pronounced by a male speaker. The ambient noise is the same as the one use to record the signal of Figure 4-4. We can observe in Figure 4-18 that the signal consists of two speech regions and three noise-only regions.

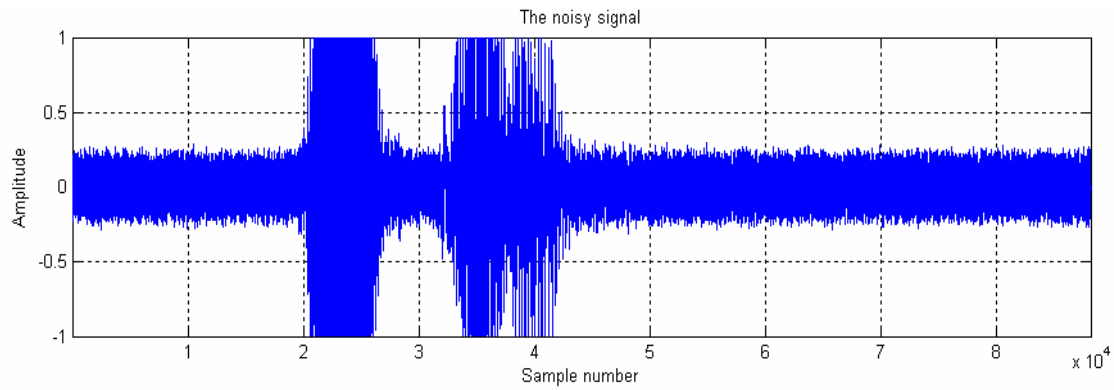


Figure 4-18 Noisy Sample Speech "They Study"

For this sentence, we consider six different filter orders: $N = 2$, $N = 8$, $N = 32$, $N = 64$, $N = 128$ and $N = 256$. The resulting waveforms are presented in Figure 4-19.

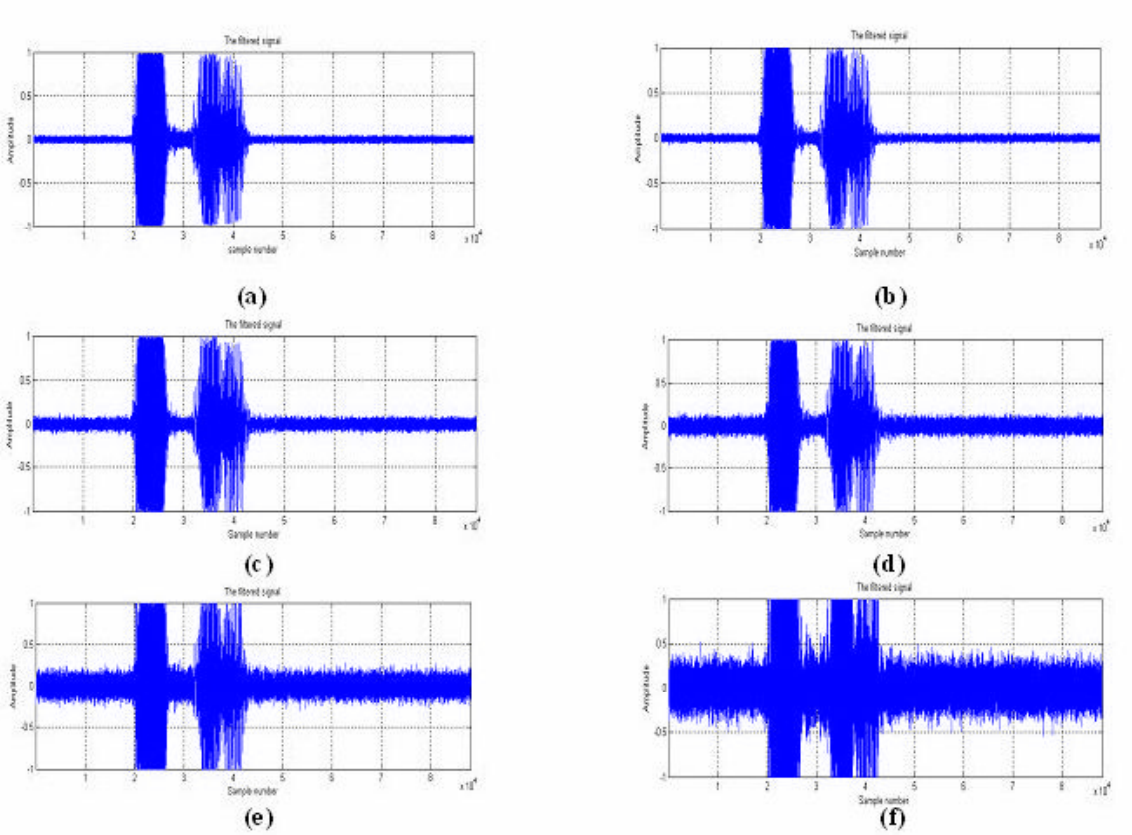


Figure 4-19 Filtered Signal for: (a) $N = 2$ (b) $N = 8$ (c) $N = 32$ (d) $N = 64$ (e) $N = 128$ (f) $N = 256$

A substantial noise reduction is observed for $N = 2$, 8, 32 and 64. However, when the order of the filter exceeds the size of the shift between windows (100 samples), the performance of the filter degrades significantly. A large number of experimentations on the size of the analysis window and on the size of the shift are collected in reference [50]. It is found that a greater noise reduction is accomplished using a second order filter with a 256 sample-length window rather than a 64-order filter with a 1064 sample-length window.

Subjective listening tests have been conducted by Miss Kaddouri [50, 51]. The first test is a speech quality test of the same type as the one that we have conducted for the clipped speech restoration. Ten people were asked to listen to noisy speech and then to filtered speech. The filter is a second order time varying filter. The listeners were asked to grade each speech heard on a scale from 1 to 5, according to how "pleasant" their listening experience was. The results are presented in Table 4-7. We observe that, according to this test, the quality of the speech has considerably improved after being filtered by the proposed time-varying filter.

	1st sample average grade	2nd sample average grade	3rd sample average grade	4th sample average grade	**Total average grade**
Noisy speech	2.00	1.90	1.90	1.80	**1.90**
Filtered speech	4.00	3.95	4.00	4.10	**4.01**

Table 4-7 Results of the Speech Quality Measurement Test [50]

The second test is an intelligibility test called Diagnosis Rhyme Test (DRT) [59]. It consists on six word pairs that differ by a single acoustic feature in an initial consonant. The word pairs are chosen to evaluate the phonetic characteristics listed in Table 4-8.

Characteristics	**Description**	**Example**
Voicing	Voiced-unvoiced	***Dense-tense***
Nasality	Nasal-Oral	***Need-deed***
Sustention	Sustained-interrupted	***Sheet-cheat***
Sibilation	Sibilated-unsibilated	***Sing-thing***
Graveness	Grave-acute	***Weed-reed***
Compactness	Compact-diffuse	***Key-tea***

Table 4-8 DRT Characteristics [59]

The listener hears six words in a row, one from each category. In an answering sheet, two options are given for each word. The listener has to mark on the answering sheet which one of the two words he thinks is correct. The test has been conducted on 10 listeners and the results are presented in Table 4-9. The filter is again a second order FIR time varying filter. We see clearly that the filtering has greatly improved the intelligibility.

Finally, we see that the best filtering is obtained by a low order time varying Wiener filter. A great advantage of the time varying filter is the fact that the signal is not distorted. This is due to the fact that the filter adapts itself to the variations of the signal. A remarkable fact is that the best filtering has been obtained by a second order

FIR filter. This is essentially due to the fact that the time varying filter has a time variable gain. The gain is quite small in the noise only part, while it is much higher (≈ 1) in the speech region.

Error rate / speech	1st listener	2nd listener	3rd listener	4th listener	5th listener	6th listener	7th listener	8th listener	9th listener	10th listener	*Total*
Noisy	66.6 %	83.3 %	83.3 %	66.6 %	50%	66.6 %	66.6 %	50%	66.6 %	83.3 %	68.3 %
Filtered	16.6 %	16.6 %	16.6 %	0%	16.6 %	16.6 %	0%	16.6 %	0%	16.6 %	11.6 %

Table 4-9 Intelligibility Measurement Test Results [50]

We have described speech pre-processing, i.e. processing before feature extraction. We tried to correct most of the inconveniences that can affect speech acquired by a typical PC sound card. The next step is to extract parameters from the "cleaned" speech.

Chapter 5

Feature Extraction and Selection

At this point, the speech signal has been pre-processed and cleaned of unwanted distortion and noise. The next step is to extract from this "raw" signal a set of parameters that are as informative as possible on the phones or phonemes being pronounced by the speaker whereas we should eliminate all redundant information and also unneeded information (about the speaker, his/her emotive state, etc) that might hinder the recognition process. We have seen that speech is a slow varying non stationary random process. So, we are going to analyse the speech signal using time frames of few milliseconds duration. In fact, we are going to use the same framing as the one described in Figure 4-15, i.e. frames 256 sample-long translated by 100 samples each time. If the sampling rate is 16 kHz, this represents frames taken every 6.25 milliseconds, each one having a duration of 16 milliseconds. We are going to analysis three different parametric representations: the linear prediction coefficients (LPC), the partial correlation coefficients (PARCOR) and the Mel frequency Cepstral coefficients (MFCC). These three sets are going to be compared from a statistical point of view and the best one will be selected for the final recognition system and for its training.

5.1 Linear Prediction

Linear prediction is a technique for the representation of speech segments using the model of production presented in chapter 1. We have seen that the speech production system can be modelled as slowly time varying linear filter $H(z)$. There exist two different formulation of linear prediction: The stationary or autocorrelation formulation [57, 58, 44, 45] and the non-stationary or covariance formulation [3, 56, 104].

5.1.1 The Autocorrelation Formulation

We have seen in chapter 3 that equation (3.1) allows us to predict a speech sample by a linear combination of the p previous samples. This formulation forms the basis of the linear prediction method. In chapter 4, we have also considered autoregressive modeling of stationary random processes. Consider the following all pole system driven by a white noise $w(n)$:

$$H(z) = \frac{b_0}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{5.1}$$

The output of the above system is:

$$x(n) = \sum_{k=1}^{p} a_k x(n-k) + b_0 w(n) \tag{5.2}$$

It consists of a sum of two signals. A linearly predicted signal:

$$\hat{x}(n) = \sum_{k=1}^{p} a_k x(n-k) \tag{5.3}$$

and a white process $b_0 w(n)$. If the process has a small variance, then, we can say that the signal $x(n)$ can be linearly predicted from its p past values in the mean square sense. As a result, we can represent the properties of the signal using the set of parameters $\{a_k\ ; k = 1,..., p\}$ that minimizes the following mean square error:

$$E = E\left[\left(x(n) - \hat{x}(n) \right)^2 \right] \tag{5.4}$$

This problem has already been seen in chapter 4, the solution is provided by the Yule-Walker equations (4.24) which are repeated below:

$$\begin{pmatrix} r_{xx}(1) \\ \vdots \\ r_{xx}(p) \end{pmatrix} = \begin{pmatrix} r_{xx}(0) & \cdots & r_{xx}(p-1) \\ \vdots & \vdots & \vdots \\ r_{xx}(p-1) & \cdots & r_{xx}(0) \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_p \end{pmatrix} \tag{5.5}$$

or $$\mathbf{r} = \mathbf{R}\mathbf{a}$$

where $r_{xx}(k)$ is the autocorrelation function of the signal $x(n)$. This autocorrelation can be computed using the unbiased formula given by equation (4.26), however, if one minimizes directly (5.4) in the least square sense, using a windowed signal, we find as result the biased value:

$$r_{xx}(k) = \frac{1}{L} \sum_{n=-\infty}^{+\infty} \tilde{x}(n)\tilde{x}(n+k) \tag{5.6}$$

where $\tilde{x}(n)$ is the signal $x(n)$ multiplied by a finite time window of length L. Equation (5.5) can be solved using a very efficient algorithm, the Levinson-Durbin algorithm [86]. The Levinson-Durbin algorithm is based on the fact that the matrix **R** is a symmetric Toeplitz matrix [86]. It is implemented using the following pseudo-code. The superscripts indicate an iteration number.

$$E^{(0)} = r_{xx}(0)$$

for $i = 1,2,\ldots, p$

$$k_i = \left(r_{xx}(i) - \sum_{j=1}^{i-1} a_j^{(i-1)} r(i-j) \right) \Big/ E^{(i-1)}$$

$$a_i^{(i)} = k_i$$

if $i > 1$ then

for $j = 1,2,\ldots,i-1$

$$a_j^{(i)} = a_j^{(i-1)} - k_i a_{i-j}^{(i-1)}$$

end for

end if

$$E^{(i)} = (1 - k_i^2) E^{(i-1)}$$

end

$$a_j = a_j^{(p)} \qquad j = 1,2,\ldots, p$$

A particularity of the Levinson-Durbin algorithm is the fact that the signal is represented by a succession of models of increasing orders, starting from zero up to p. A by-product of the algorithm is the set of coefficients k_i. These coefficients are called the PARCOR (partial correlation) coefficients by Itakura and Saito [44, 45]. The PARCOR coefficients are also called "reflection coefficients" and they have a direct relationship with the implementation of the speech model $H(z)$ by the lattice structure shown in Figure 5-1.

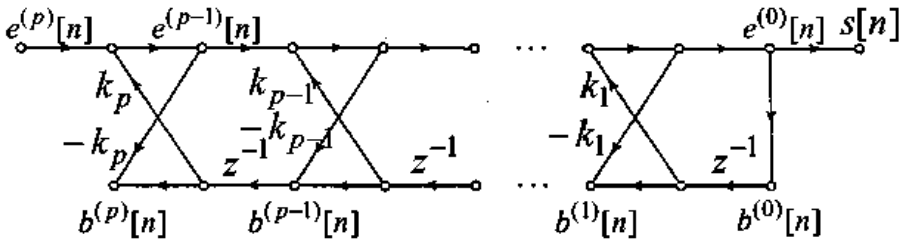


Figure 5-1 Lattice Structure of the Vocal Tract Model $H(z)$

Another property of the refection coefficients is the fact that they are bounded by one: $|k_i| < 1$. It can be shown that this implies that the obtained filter $H(z)$ is always stable. In conclusion, the particular speech segment (corresponding to the finite time window) can be represented by either the set of p prediction coefficients a_i or the set of p PARCOR coefficients k_i.

5.1.2 The Covariance Formulation

For the covariance formulation, we assume that the speech segment is voiced and let us place the time origin at the beginning of the pitch period. Let N_T the number of samples in a pitch period. If we assume that the production model has poles and zeroes (general autoregressive, moving average model, ARMA), then the input output relationship of the production model satisfies equation (3.1) which is repeated below:

$$x(n) = \sum_{k=1}^{p} a_k x(n-k) + \sum_{k=0}^{p} b_k u(n-k) \tag{5.7}$$

where the excitation $u(n)$ take the value of one at $n = 0$ and zero up to N_T. Assuming zero initial conditions and a window of size $N+1$, we can write the following set of equations:

$$\begin{aligned}
x(0) &= b_0 \\
x(1) &= b_1 + a_1 x(0) \\
x(2) &= b_2 + a_1 x(1) + a_2 x(0) \\
&\vdots \\
x(p) &= b_p + a_1 x(p-1) + a_2 x(p-2) + \cdots + a_p x(0) \\
x(p+1) &= a_1 x(p) + a_2 x(p-1) + \cdots + a_p x(1) \\
&\vdots \\
x(N) &= a_1 x(N-1) + a_2 x(N-2) + \cdots + a_p x(N-p)
\end{aligned} \tag{5.8}$$

Introducing the following notation:

$$\begin{aligned}
\mathbf{x}_1 &= \begin{pmatrix} x(0) & x(1) & x(2) & \cdots & x(p) \end{pmatrix}^T \\
\mathbf{x}_2 &= \begin{pmatrix} x(p+1) & \cdots & x(N) \end{pmatrix}^T \\
\mathbf{b} &= \begin{pmatrix} b_0 & b_1 & \cdots & b_p \end{pmatrix}^T \\
\mathbf{a} &= \begin{pmatrix} a_1 & a_2 & \cdots & a_p \end{pmatrix}^T
\end{aligned}$$

We can re-express the set of equations (5.8) in matrix form. It provides:

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{L} \\ \mathbf{0} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ \mathbf{a} \end{pmatrix} \tag{5.9}$$

where:

$$\mathbf{L} = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ x(0) & 0 & \cdots & 0 \\ x(1) & x(0) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x(p-1) & x(p-2) & \cdots & x(0) \end{pmatrix}$$

and

$$\mathbf{X} = \begin{pmatrix} x(p) & x(p-1) & \cdots & x(1) \\ x(p+1) & x(p) & \cdots & x(2) \\ \vdots & \vdots & \ddots & \vdots \\ x(N-1) & x(N-2) & \cdots & x(N-p) \end{pmatrix}$$

The least square solution of the above set of equations is given by:

$$\begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{L}^T & \mathbf{X}^T \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{I} & 0 \\ \mathbf{L}^T & \mathbf{X}^T \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{L} \\ \mathbf{0} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ \mathbf{a} \end{pmatrix} \tag{5.10}$$

Finally, we find:

$$\mathbf{X}^T \mathbf{x}_2 = \mathbf{X}^T \mathbf{X} \mathbf{a} \tag{5.11}$$

$$\mathbf{b} = \mathbf{x}_1 - \mathbf{L}\,\mathbf{a} \tag{5.12}$$

The elements of the $p \times p$ matrix $\mathbf{X}^T\mathbf{X}$ are denoted $\phi(i,k)$ and are called covariance coefficients. Their value is given by

$$\phi(i,k) = \sum_{n=p+1}^{N} x(n-i)x(n-k) \tag{5.13}$$

and the elements of the vector $\mathbf{X}^T\mathbf{x}_2$ are actually given by $\phi(i,0)$.

The set of parameters a_k, solution of the system of equations (5.11), is also called the set of prediction coefficients. The matrix $\mathbf{X}^T\mathbf{X}$, although symmetric, lacks the Toeplitz structure of the matrix $\mathbf{R}$ of equation (5.5). Nevertheless, it is usually positive definite. It can be solved by the Cholesky decomposition method [86], which is quite efficient. We note also that the covariance method is in fact a pole and zero representation of the speech production model $H(z)$. In reference [17], Chandra and Lin show that if the interval of analysis is smaller than a pitch period, the covariance method is more precise. If the window is much longer than the pitch period, the two methods become similar. Therefore, we are going to use the autocorrelation formulation in the remaining part of our work.

5.2 The Mel Frequency Cepstral Coefficients

The Mel Frequency Cepstral Coefficients (MFCC) is the most commonly used set of parameters in modern speech recognition systems [16, 26, 63]. This set of features is based on an auditory model of frequency sensation by listeners. Measurements have shown that human listeners cannot distinguish two tones if they are separated by less than a bandwidth of about 20% of the frequency of one the tones [106]. Other measurements on human perception of tones show also that our perception of frequency follows a logarithmic scale. The perceived "frequency" is called *pitch*[5] [85] and is measured in mels. The following equation is an empirical relationship between frequencies and mels [85].

$$\text{Pitch in mels} = 1127 \ln\left(1 + {}^{f}\!/_{700}\right) \tag{5.14}$$

The mel scale is calibrated by assigning the value of 1000 mels to a frequency of 1000 Hz. The concept of mel scale is closely related to the concept of critical frequency bands.

Another important notion used in the MFCC's is the concept of cepstrum [12]. The word "cepstrum" is an anagram of the word spectrum. It has been coined by Bogert [12] in 1963. This notion has been later generalized by Oppenheim et al. [64, 65] to the concept of "homomorphic" processing of signals. In speech recognition, the original definition is sufficient. It is usually defined as real cepstrum. Given a windowed segment of signal $x(n)$, the real cepstrum $c(n)$ is defined as the inverse discrete Fourier transform (IDFT) of the logarithm of the modulus of the discrete Fourier transform (DFT) of the signal $x(n)$.

$$c(n) = \text{IDFT}\left[\log\left|\text{DFT}\left(x(n)\right)\right|\right] \tag{5.15}$$

The index axis of the cepstrum is called the "quefrency" axis. If the signal $x(n)$ has been produced by the convolution of a periodic signal with some impulse response, the cepstrum is going to present at low quefrencies the transform of the impulse response and at the higher quefrencies the transform of the excitation. We can separate the two signals (deconvolve) by eliminating the high quefrency terms. This operation is called "liftering" (for filtering) and the window is called a "lifter" (for filter). If this operation is applied to the speech signal, we can eliminate the glottal response, which is

[5] It is a subjective attribute of periodic sounds. It should not be confused with the previously defined pitch frequency of voiced speech.

speaker dependent, and keep only the vocal tract response, which depends on phones being pronounced. $l_1(n)$ and $l_2(n)$ are typical lifters used in speech recognition.

$$l_1(n) = \begin{cases} 1 & n = 0,1,\ldots,L-1 \\ 0 & elsewhere \end{cases} \tag{5.16}$$

$$l_2(n) = \begin{cases} 1 + \dfrac{L-1}{2}\sin\left(\dfrac{\pi n}{L-1}\right) & n = 0,1,\ldots,L-1 \\ 0 & elsewhere \end{cases} \tag{5.17}$$

L being smaller than the size of the window.

The Mel Frequency Cepstral coefficients (MFCC) is a parametric representation of speech that utilizes a filter bank with bandwidths that are constant on a mel scale and also liftering in order to eliminate the effect of the excitation on the vocal tract response [23]. For a 4 kHz bandwidth, approximately 20 filters are needed. In practice, a short time DFT is computed, resulting in a spectrum $X_n(k)$ for the frame positioned at sample n.

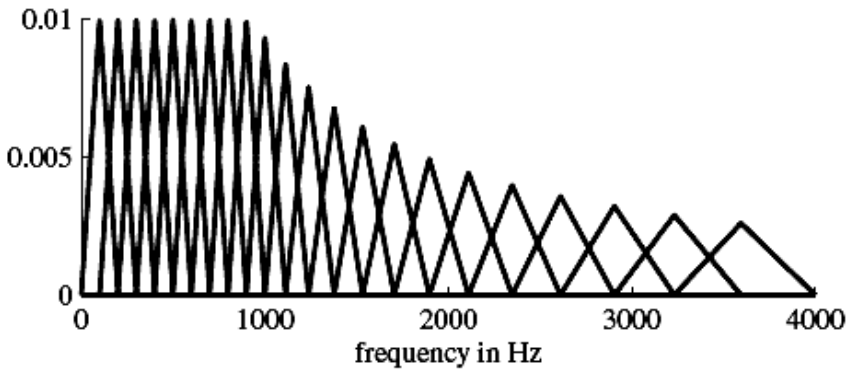


Figure 5-2 Weighting Functions for Mel-Frequency Filter Bank

The DFT values are then grouped together in critical bands and weighted using the triangular functions shown in Figure 5-2. We can remark that the bandwidth are constant below 1 kHz and then increase exponentially up to 4kHz, resulting in 22 analysis filters. So, the mel frequency power spectrum is defined as the weighted power out of the analysis filters for r = 1, ..., R.

$$MF_n[r] = \frac{1}{A_r}\sum_{k=L_r}^{U_r} \left| V_r(k) X_n(k) \right|^2 \tag{5.18}$$

where $V_r(k)$ is the r^{th} triangular weighting function spanning the sample frequency index from L_r to U_r and A_r is a normalizing constant.

$$A_r = \sum_{k=L_r}^{U_r} |V_r(k)|^2 \tag{5.19}$$

This normalization is built into the filters shown in Figure 5-2. It is needed, so that a flat Fourier spectrum will produce a flat mel spectrum. The final MFCC coefficients are then computed using a discrete cosine transform (DCT) of the logarithm of the filter output. The DCT is used instead of an IDFT because we are dealing with a real symmetric spectrum.

$$c_n(m) = \frac{1}{R}\sum_{r=1}^{R} \log\left(MF_n[r]\right) \cos\left[\frac{2\pi}{R}\left(r+\frac{1}{2}\right)m\right] \tag{5.20}$$

At this level, a liftering is performed since we compute only N_{MFCC} coefficients where $N_{MFCC} < R$. Typical values are $N_{MFCC} = 13$ and $R = 22$. In some applications, we compute also first difference coefficients $\Delta c_n(m)$ (to approximate the derivative) and second difference coefficients $\Delta\Delta c_n(m)$ (to approximate the acceleration). These difference coefficients are useful to characterize transient phones like plosives but they increase the dimension of the feature space (from 13 to 39). In the following section, we are going to compare three set of features from the statistical point of view. The three sets are: the set of linear prediction coefficients $\{a_1, a_2, ..., a_{16}\}$ from the autocorrelation formulation, the set of PARCOR coefficients $\{k_1, k_2, ..., k_{16}\}$ from the same formulation and the set of MFCC coefficients $\{c(1), c(2), ..., c(13)\}$.

5.3 Statistical Comparison of Features

5.3.1 The Normal Density

In our model of speech recognition, we assume that the phones cluster in different classes. The basic assumption is that the parameters extracted from a given speech segment are members of a given statistical population, that is, they are realizations of random vectors having a given probability density function (PDF). It is quite common to assume that the pdf is unimodal. Therefore, it is convenient to assume that the features derive from a multivariate Gaussian density (Normal density) [49].

$$f_{\mathbf{x}|w_i}(\mathbf{x}\,|\,w_i) = N\left(\mathbf{x}, \mathbf{m}_i, \mathbf{S}_i\right) = \frac{1}{(2\pi)^{\frac{p}{2}}\,|\mathbf{S}_i|^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{m}_i)^T\,\mathbf{S}_i^{-1}(\mathbf{x}-\mathbf{m}_i)\right] \tag{5.21}$$

Equation (5.21) describes a pdf for a normal density for a p-dimensional vector $\mathbf{x}$ coming from the class with label w_i, with mean $\mathbf{m}_i$ and covariance matrix $\mathbf{S}_i$. The constant pdf surfaces are hyper-ellipsoids described by equation (5.22).

$$d^2(\mathbf{x}, \mathbf{m}_i) = (\mathbf{x} - \mathbf{m}_i)^T \mathbf{S}_i^{-1} (\mathbf{x} - \mathbf{m}_i) = c^2 \tag{5.22}$$

$d^2(\mathbf{x},\mathbf{m}_i)$ is called the Mahalanobis distance between $\mathbf{x}$ and $\mathbf{m}_i$. The principal axes of the hyper-ellipsoid are given by the eigenvectors of $\mathbf{S}_i$ and their lengths are proportional to the square root of the eigenvalues of $\mathbf{S}_i$. We will show that the Mahalanobis distance has a chi-square distribution with p degrees of freedom. So, if $c^2 = \chi_p^2(\alpha)$, where $\chi_p^2(\alpha)$ is the (100α) percentile (quantile) of a chi-square distribution with p degrees of freedom, then we have 100α % of chance to find a vector $\mathbf{x}$ belonging to the class w_i inside the hyper-ellipsoid defined by (5.22). Therefore, if the distributions of all classes are normal, the samples will have a tendency to cluster in hyper-ellipsoids. We are going to verify which set of features obeys (as much as possible) a normal law.

5.3.2 Assessing the Assumption of Normality

When we deal with one-dimensional data, a commonly used test for normality of the data is the QQ (quantile-quantile) plot [49]. Assume we have collected N random data $X_1, X_2, ..., X_N$. The order statistics $X_{[1],N}, X_{[2],N}, ..., X_{[N],N}$ is a rearrangement of the initial data in increasing order. If we assume that the data came from a population having a given pdf $f_X(x)$, then it is shown [11] that $X_{[j],N}$ is the "sample p-quantile" where j is the smallest integer greater than or equal to Np. The p-quantile θ_p is defined as:

$$p = \int_{-\infty}^{\theta_p} f_X(x)dx \tag{5.23}$$

Let θ_p , $p = (j - 0.5)/N$, $j = 1, ...,N$, be N successive quantiles from the above defined distribution. A QQ plot is a plot of the sample p-quantiles against the quantiles θ_p. If the data are effectively drawn from the same type of distribution, then the plot will be a straight line. We can measure the degree of linearity by computing the correlation coefficient between the data. The data will be Gaussian (Normal) if the p-quantiles are drawn from a standard normal distribution.

$$p = \int_{-\infty}^{\theta_p} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \tag{5.24}$$

When the collected data are multidimensional, the above test can be used to examine whether the marginal distributions are normal or not. However, this does not imply that the overall distribution is Gaussian if all the marginal distributions are normal. A better test is the Chi-square plot [49]. We have seen that the Mahalanobis distance d^2 has a chi-square distribution. So, if we plot sorted Mahalanobis distances

versus chi-square quantiles, then we can assess whether the distribution of d^2 is chi-square or not. This will allow us to deduce that the distribution of the vectors is normal if the chi-square plot is linear.

We are going to analyze three different parametric representation of speech. The speech signal is coming from two different locutors: one male ('Kader') and one female ('Aicha') sampled at 16 kHz. We use 256 sample wide Hamming windows shifted by 100 samples at each frame. The sets are:

- 16 LPC coefficients issued from the autocorrelation formulation. We also call them the "a parameters".
- 16 PARCOR coefficients issued from the same formulation. We also call them "k parameters".
- 13 MFCC coefficients.

Since our data are multidimensional, we are going to use Chi-square plots. The first step in our analysis is the manual extraction of speech segments corresponding to the different phonemes of Table A-1. For each one of these segments, we extract different sets of features ("a", "k" and MFCC").

Figure 5-3 and Figure 5-4 show two Chi-square plots relating to MFCC parameters. Figure 5-3 concerns 727 frames of different utterances of the phoneme /oo/ pronounced by Kader while Figure 5-4 concerns 723 frames of utterances of the phoneme /aa/ pronounced by Aicha. We can notice that the plots are quite linear, especially for values of squared Mahalanobis distance not exceeding 30. This tendency is confirmed by the correlation coefficients, which have respectively the values: $r = 0.9892$ for the phoneme /oo/ pronounced by Kader and $r = 0.9829$ for /aa/ pronounced by Aicha.

Figure 5-5 and Figure 5-6 show two Chi-square plots concerning LPC "a" parameters. Figure 5-5 is about 581 frames of utterances of the phoneme /aa/ pronounced by Kader while Figure 5-6 is about 802 frames of utterances of the phoneme /oo/ pronounced by Aicha. It is visible that the plots are not as linear as the ones pertaining to MFCC parameters. The correlation coefficients are respectively $r = 0.9076$ for the phoneme /aa/ pronounced by Kader and $r = 0.7978$ for /oo/ pronounced by Aicha.

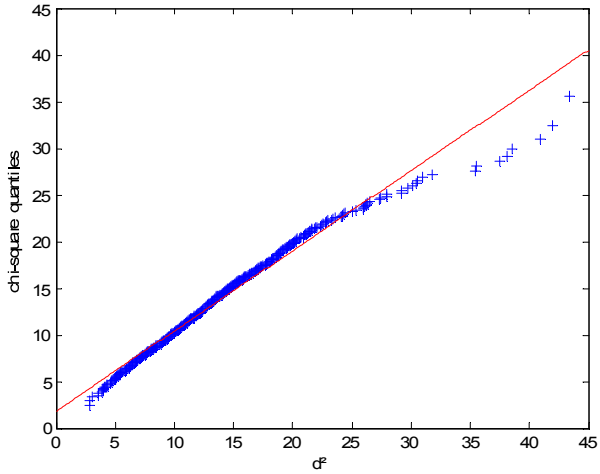


Figure 5-3 Chi-Square Plot for MFCC of /oo/ by Kader

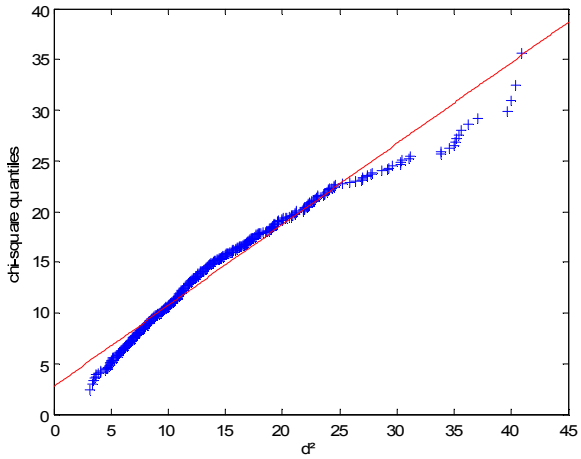


Figure 5-4 Chi-Square Plot for MFCC of /aa/ by Aicha

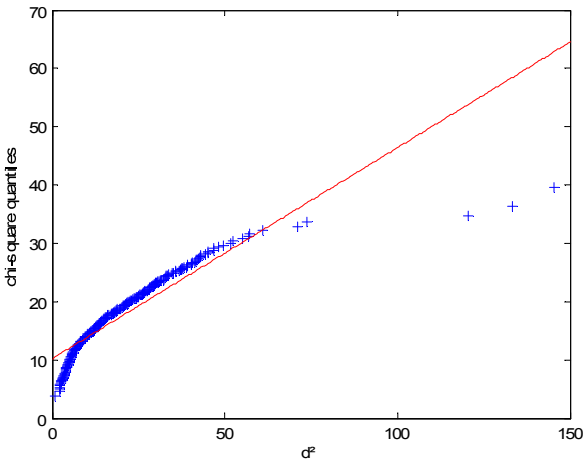


Figure 5-5 Chi-Square Plot for "a" parameters of /aa/ by Kader

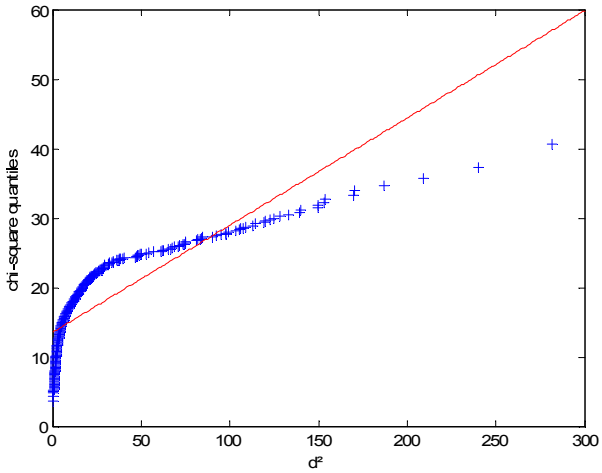


Figure 5-6 Chi-Square Plot for "a" parameters of /oo/ by Aicha

Figure 5-7 and Figure 5-8 show two Chi-square plots pertaining to PARCOR "k" parameters. Figure 5-7 concerns 581 frames of utterances of the phoneme /aa/ pronounced by Kader while Figure 5-8 concerns 799 frames of utterances of the phoneme /aa/ pronounced by Aicha. We see that the plots are not linear at all. This is confirmed by the correlation coefficients which are respectively $r = 0.5850$ for the phoneme /aa/ pronounced by Kader and $r = 0.5628$ for /aa/ pronounced by Aicha.

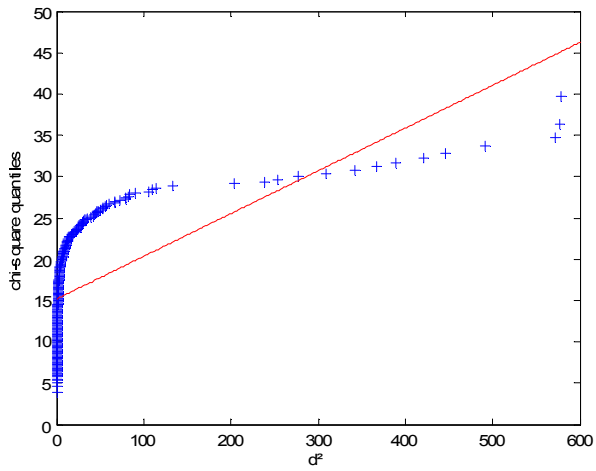


Figure 5-7 Chi-Square Plot for "k" parameters of /aa/ by Kader

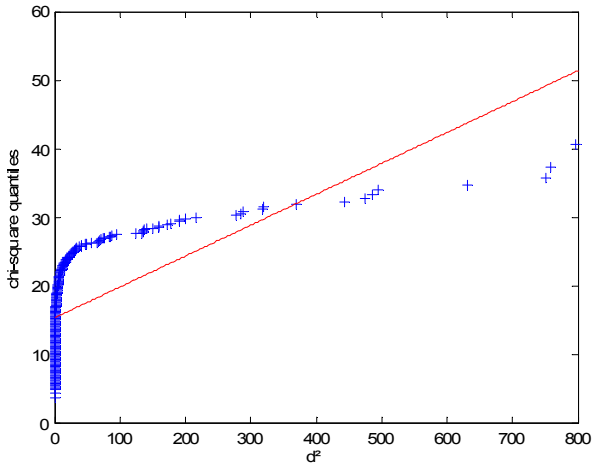


Figure 5-8 Chi-Square Plot for "k" parameters of /aa/ by Aicha

We have repeated the above tests for most sustainable phonemes of Table A-1. The results are summarized in Table 5-1 for the male speaker (Kader) and in Table 5-2 for the female one (Aicha).

Kader	/aa/	/oo/	/ii/	/rr/	/ss/	/ff/	/mm/
MFCC	0.9850	0.9892	0.9853	0.9699	0.9581	0.9712	0.9778
PARCOR	0.5850	0.5748	0.5783	0.5230	0.5177	0.5623	0.5576
LPC	0.9076	0.8925	0.8991	0.9020	0.9102	0.8925	0.8953

Table 5-1 Correlation Coefficients for Kader

Aicha	/aa/	/oo/	/ii/	/rr/	/ss/	/ff/	/mm/
MFCC	0.9829	0.9725	0.9679	0.9564	0.9600	0.9501	0.9550
PARCOR	0.5628	0.5542	0.5538	0.5378	0.5221	0.5012	0.5566
LPC	0.8022	0.7978	0.7899	0.7543	0.7830	0.7692	0.7669

Table 5-2 Correlation Coefficients for Aicha

From the above tables, it is apparent that the MFCC exhibit a more pronounced Gaussian character than the other two sets of features. In fact, the Gaussian behavior is

more evident for smaller values of the Mahalanobis distance to the mean of the class. Considering only the Gaussian behavior, we should select the MFCC set of parameters.

5.3.3 Discriminant Analysis

Another factor to take into consideration in the choice of a set of features is the separation of the different clusters in the feature space. Ideally, the clusters should have very small variances and they should be widely separated. A graphical tool that is very useful is a scatter plot of the data. However, it can be used only in two-dimensional spaces. We have seen that the parameters under study are 13-dimensional vectors for the MFCC, 16-dimensional vectors for the LPC coefficients and 16-dimensional vectors for the PARCOR coefficients. In order to have significant scatter plot, we should define a p to 2 mapping. The mapping that we are going to use must retain as much properties as possible. Among all possible mappings, we are going to select one such that the classes are going to be as separated as possible in the range space.

Introduced by Fisher [28, 29] and later developed by Wilks [101], statistical discriminant analysis is a way of transforming a set of p-dimensional vector observations into a set of m-dimensional vectors, $m < p$. The transformed vectors have the property that most of the information of the original set is preserved in the lower dimensional space. These transformations are commonly used in statistical pattern recognition [35, 88] We assume that we have N p-dimensional observations belonging to M different classes, that is, N_1 vectors in class w_1, N_2 in w_2, ..., N_M in w_M, along with $N_1 + N_2 + ... + N_M = M$. For each class, we can define a class center, i.e. the class mean using (5.25) and a scatter matrix, which is nothing but the class covariance matrix using (5.26).

$$\mathbf{m}_i = \frac{1}{N_i}\sum_{k=1}^{N_i}\mathbf{x}_k \qquad ; \quad \mathbf{x}_k \in w_i \tag{5.25}$$

$$\mathbf{W}_i = \frac{1}{N_i - 1}\sum_{k=1}^{N_i}\left(\mathbf{x}_k - \mathbf{m}_i\right)\left(\mathbf{x}_k - \mathbf{m}_i\right)^T \qquad ; \quad \mathbf{x}_k \in w_i \tag{5.26}$$

The matrix $\mathbf{W}_i$ gives an indication of the distribution of data from class w_i around its mean. However, this is "local" information that pertains only to class w_i. More global information is provided by the *within* scatter matrix:

$$\mathbf{W} = \sum_{i=1}^{M}\frac{N_i}{M}\mathbf{W}_i \tag{5.27}$$

The within scatter matrix provides information about how data cluster around the different means: The smaller its eigenvalues, the more concentrated the data will be around its means. We can also define an overall mean:

$$\mathbf{m}_0 = \sum_{i=1}^{M} \frac{N_i}{M} \mathbf{m}_i \tag{5.28}$$

Information about the average separation of classes is given by the *between* scatter matrix:

$$\mathbf{B} = \sum_{i=1}^{M} \frac{N_i}{M} (\mathbf{m}_i - \mathbf{m}_0)(\mathbf{m}_i - \mathbf{m}_0)^T \tag{5.29}$$

B being the sum of M rank one matrices will automatically be singular if M is smaller than the dimension of the observation space. Classes will be widely separated and the data will be tidily clustered around their means if the following criterion is maximized:

$$J = tr\left[\mathbf{W}^{-1}\mathbf{B}\right] \tag{5.30}$$

where $tr[\mathbf{A}]$ stands for trace of **A**.

Let us define a transformation matrix **A** from the p-dimensional space to an m-dimensional. The within and the between scatter matrices are going to become:

$$\mathbf{B}_{(m)} = \mathbf{A}^T \mathbf{B} \mathbf{A} \quad \text{and} \quad \mathbf{W}_{(m)} = \mathbf{A}^T \mathbf{W} \mathbf{A} \tag{5.31}$$

It is shown that the optimum transformation matrix, that is, the matrix **A** that maximizes J in the m-dimensional space is given by the matrix whose columns are the eigenvectors corresponding to the largest eigenvalues of the following generalized eigen-problem:

$$\mathbf{B}\mathbf{x} = \lambda \mathbf{W}\mathbf{x} \tag{5.32}$$

Figure 5-9, Figure 5-10, Figure 5-11and Figure 5-12 represent two-dimensional scatter plots of measured data from three vowels pronounced by Kader and Aicha. The axes of the two-dimensional space are the eigenvectors corresponding to the largest eigenvalues of (5.32) computed in the space of LPC parameters (16-dimensional) and the space of MFCC (13-dimensional). Scatter plots for the PARCOR coefficients are not shown because it is impossible to distinguish the three classes in the scatter plots. Figure 5-9 and Figure 5-11 represent the scatter plots for the LPC coefficients while Figure 5-10 and Figure 5-12 represent scatter plots for the MFCC parameters. It is apparent that the MFCC coefficients provide a better class separation. Furthermore, we

notice the typical ellipsoidal shape of the different clusters, which confirms the Gaussian nature of the MFCC parameters.

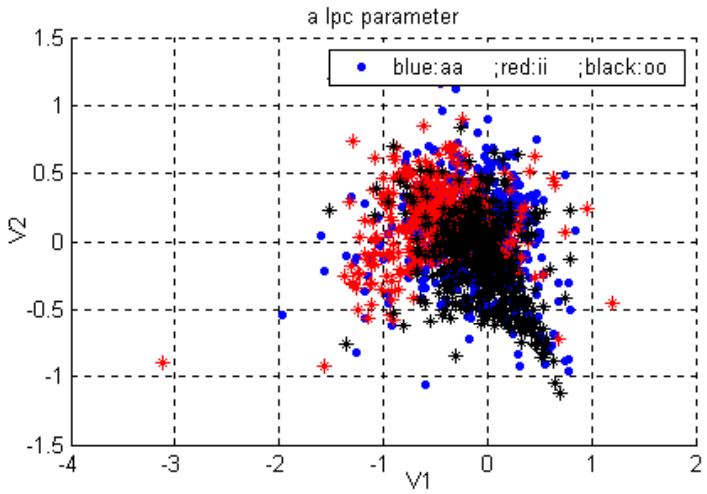


Figure 5-9 Scatter Plot of LPC parameters by Aicha

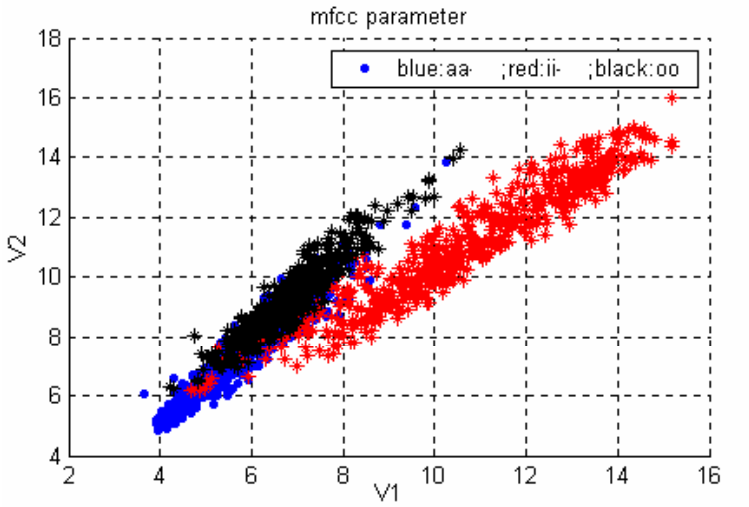


Figure 5-10 Scatter Plot of MFCC Parameters by Aicha

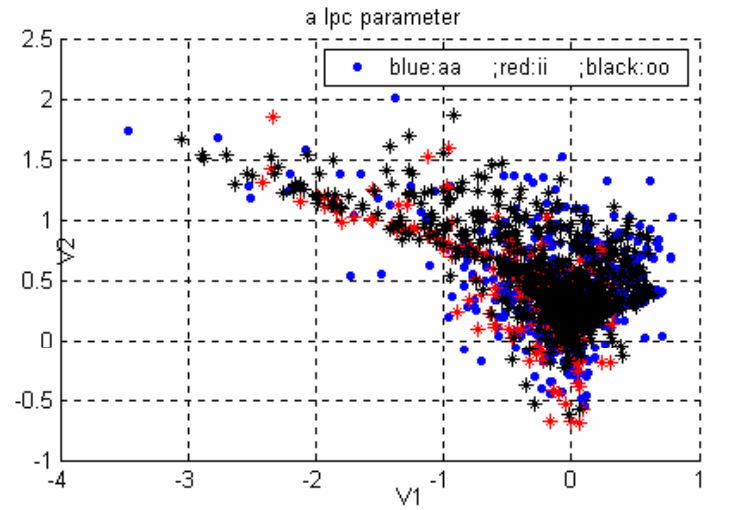


Figure 5-11 Scatter Plot of LPC parameters by Kader

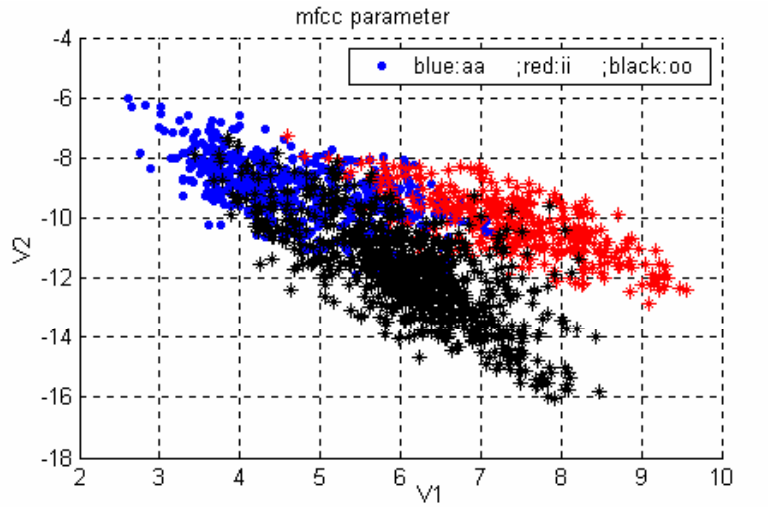


Figure 5-12 Scatter Plot of MFCC Parameters by Kader

The scatter plots indicate that the MFCC parameters form a better set of features than the LPC parameters. This is confirmed by the value of $J = tr[\mathbf{W}^{-1}\mathbf{B}]$ computed for both speakers and both set of parameters. The separability of the set of MFCC is much higher than the one provided by the LPC parameters.

$tr[\mathbf{W}^{-1}\mathbf{B}]$	LPC	MFCC
Aicha	0.00050534	0.0029941
Kader	0.0006328	0.0029114

Table 5-3 *J* Criterion Value for LPC and MFCC Parameters

The statistical analysis indicates clearly that we should use the MFCC parameters to represent speech segments for the recognition stage of our system.

Chapter 6

Continuous Speech Recognition

Speech recognition is a subdivision of the more general theory of pattern recognition. In our work, we are going to use results from statistical pattern recognition [35, 88]. In the present chapter, we are going to compare two different techniques for speech recognition. The first one uses the classical hidden Markov models (HMM) [76] and will be used as a benchmark against which we are going to compare our stack search method [20].

6.1 Statistical Pattern Recognition

Statistical pattern recognition [35, 88] is essentially a method of classification. Let us assume we are given a measurement vector **x**. This vector represents features from some pattern w_i. If we know that there exists M pattern classes $\{w_1, w_2, \ldots, w_M\}$, the job of the pattern recognizer (classifier) is the identification of the class w_i according to some optimality criterion. The usual (and most widely used) criterion of optimality is the minimum probability of error. It is shown that minimizing the probability of error is equivalent to maximizing the "A posteriori" probability. So, the MAP (Maximum A posteriori Probability) rule is:

$$\text{select } w_i \text{ such that:} \quad P\left[w_i \middle| \mathbf{x}\right] \text{maximum}; i \in \{1,2,\ldots,M\} \tag{6.1}$$

where $P[w_i|\mathbf{x}]$ is the a posteriori probability of the class w_i given the measurement **x**. If the patterns are generating continuous random variables, application of the Bayes rule generates the following decision function:

$$d_i(\mathbf{x}) = P\left[w_i\right] f_{\mathbf{X}|w_i}(\mathbf{x} \mid w_i) \tag{6.2}$$

and the decision rule becomes:

$$\text{select } w_i \text{ such that:} \quad d_i(\mathbf{x}) \text{ maximum}; i \in \{1,2,\ldots,M\} \tag{6.3}$$

$f_{\mathbf{X}|w_i}(\mathbf{x} \mid w_i)$ is the conditional density of **x** given that it comes from the class w_i. It is also called the "likelihood" function. $P[w_i]$ is the a priori probability of the class w_i. If all classes are equiprobable, the decision rule is based only on the likelihood function. It is called a ML or maximum likelihood decision rule. The decision rules divide the

observation space into M non-intersecting regions $\mathcal{D}_i$. The probability of error can be evaluated as:

$$P[E] = 1 - \sum_{i=1}^{M} P[w_i] \int_{\mathcal{D}_i} f_{\mathbf{X}|w_i}(\mathbf{x} \mid w_i) d\mathbf{x} \tag{6.4}$$

Equation (6.4) can be very hard to compute and is usually evaluated statistically. It is an important figure of merit of a recognition algorithm. However, the presented theory is valid for static patterns. This is not the case of speech. We have seen in chapter two that speech can be represented as a succession of phones. When we analyze a speech segment using translated overlapping windows, the sequence of measured data is going to trace a path in the feature space. Figure 6-1 shows an example of a given path in a two dimensional space which shows that the phone corresponding to class w_1 is pronounced during a given time. Then there is a transition to another phone and the signal remains in class w_2, etc. A solution to word recognition would be to try to recognize every segment independently and then assemble the result of the recognition to recognize the word or phrase being pronounced. The drawback of this method is the fact that some segments might be in error and that the number of segments corresponding to a particular phone is variable. We can remark this variation in Figure 6-2 and Figure 6-3, where the same locutor pronounces the same word, but in different disposition. This approach does not give satisfactory results. We obtain better results by following the segment recognizer with some smoothing algorithm [61]. Even with this improvement, the recognition rate is not very high unless the vocabulary size is small.

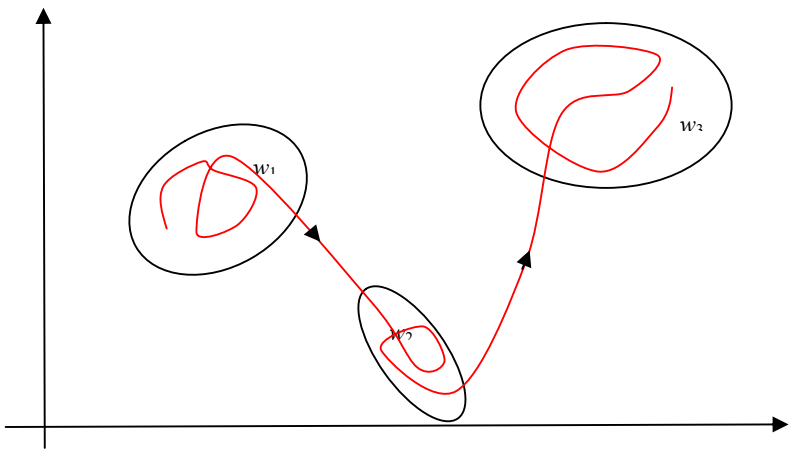


Figure 6-1 Path Followed by a Speech Signal in the Feature Space

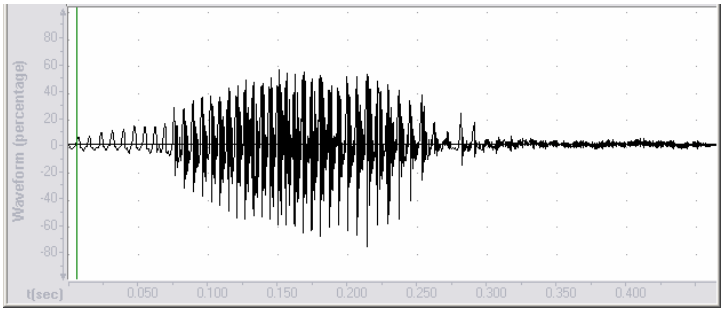


Figure 6-2 Word "mars" pronounced by Kader on day 1

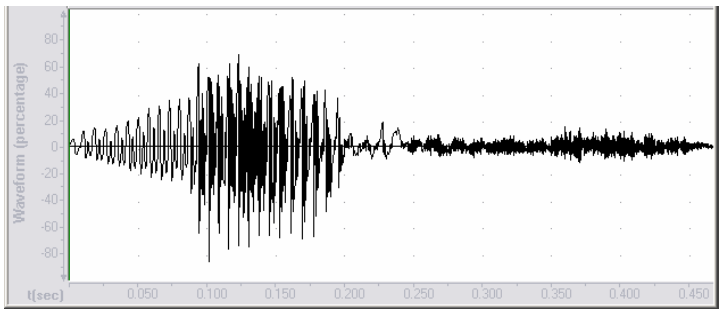


Figure 6-3 Word "mars" pronounced by Kader on day 2

All large vocabulary speech recognition systems model words as concatenation of sub-words units such as syllables or phonemes. These models allow also dynamic modelling of speech in order to take into account the variabilities described above.

6.2 Finite State Automata

Finite state automata (FSA) are commonly used for language modelling. In speech recognition, we can use FSA to model word sub-units, (phonemes or syllables), words or even phrases or complete sentences. Using FSA, we can introduce the notion of time variation in the speech segment. The raw speech signal is analyzed in frames consisting of finite time windows (usually 256 samples) translated by R samples (usually 100 samples).

Finite State Automaton Definition [53]:

A finite state automaton is an abstract machine consisting of:

1. A set of states $Q = \{I, 1, 2, \ldots, K, F\} \subset \mathbb{N}$, where I is an initial state and F is a final state. F is also referred as accepting state in the case of finite state recognizer (FSR). We can have multiples initial and final states. The state visited at frame index t will be denoted q_t.

2. A set Y of input symbols. The symbols can be discrete or continuous, scalars or vectors. The input at frame index t will be denoted y_t.
3. A set X of output symbols. The symbols can be discrete or continuous, scalars or vectors. The output at frame index t will be denoted x_t.
4. A state transition function $q_t = f(y_t, q_{t-1})$ called also the next state equation. In many cases, it is available in the structure of a table.
5. An emission function $x_t = g(q_t, q_{t-1})$ which takes the current state and the previous state and returns the output pattern x_t. This automaton is usually known as a *Mealy* FSA or branch emitting FSA. A variant of this, the *Moore* FSA, has an emission function that depends only on the current state $x_t = g(q_t)$. It is also called state emitting FSA.

We assume that the FSA is synchronous: it makes a state transition at each frame. In many cases, the FSA model is used without the input set Y. The complete machine can then be described by a single transition table or graphically, using a state diagram as shown in Figure 6-4. It represents a two state Mealy FSA with outputs $\mathbf{x}_0$ and $\mathbf{x}_1$. The same information is provided by Table 6-1.

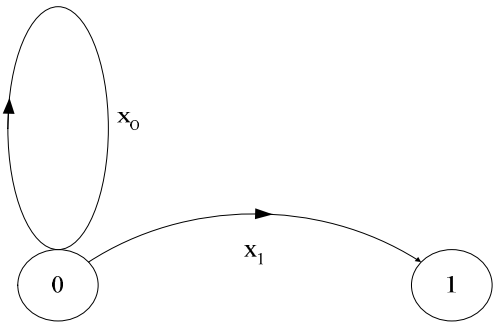


Figure 6-4 Example of a State Diagram

Present state	Next state	Branch label
0	0	x_0
0	1	x_1

Table 6-1 State Transition Table

The state diagram provides only static transition information. Better information on the time evolution of the FSA is provided by a state trellis. The state trellis is a two dimensional representation of state transitions. The horizontal axis is a time (frame) axis and the vertical axis represents the states at the given frame index.

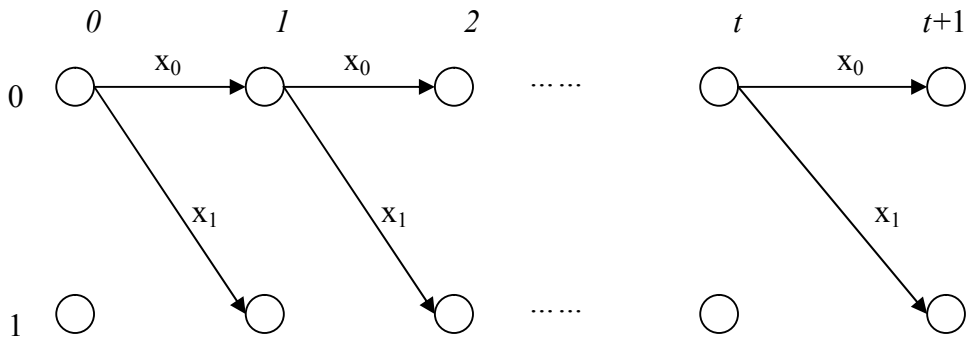


Figure 6-5 State Trellis Corresponding to Figure 6-4

A particular realization of the process described by the above state automaton will trace a path in the state trellis. FSA can be quite useful to describe speech segments. We have seen in chapter 2 that a particular speech segment can be modelled as a succession of quasi-stationary small segments, each segment corresponding to a branch label for a Mealy machine or a node (state) label for a Moore machine.

6.3 Hidden Markov Models

The FSA described above are deterministic FSA. There exist Stochastic FSA's (SFSA). An FSA is stochastic when the production function *g* (.) is not deterministic. An important class of SFSA is represented by hidden Markov models (HMM) [5, 6, 7, 8, 16, 34, 62, 67, 72, 73, 74, 76]. It is essentially a Markov chain whose states cannot be observed directly. A Markov chain is a FSA that can be described by a state diagram with branch labels being transition probabilities between states. So, an HMM will be defined by:

1. A finite set of states $Q = \{1, 2, \ldots, N\}$. The state at time *t* will be labelled as q_t.
2. The state transition probability is the probability of moving from state *i* to state *j* in one frame step.

$$a_{ij} = P\left[q_{t+1} = j \,|\, q_t = i\right] \quad ; \quad 1 \le i, j \le N \tag{6.5}$$

Using the finite memory of the Markov chain:

$$P\left[q_{t+1} = j \,|\, q_t = i; q_{t-1} = k_{t-1}; q_{t-2} = k_{t-2}; \ldots; q_0 = k_0\right] = P\left[q_{t+1} = j \,|\, q_t = i\right]$$

These probabilities can be represented by an $N \times N$ transition matrix $\mathbf{A} = \{a_{ij}\}$.

3. The initial state probabilities are defined as:

$$\pi_i = P\left[q_0 = i\right] \quad ; \quad 1 \le i \le N \tag{6.6}$$

4. To every state, at frame index t, we attach an observation $\mathbf{x}_t$ (feature vector for example) with a conditional probability (or conditional density):

$$b_j\left(\mathbf{x}_t\right) = P\left[\mathbf{x}_t \left| q_t = j \right.\right] \tag{6.7}$$

Properties 1, 2 and 3 are nothing but the definition of a Markov chain, while the last property relates the observations to the Markov chain. It is evident, from the definition, that the states of the Markov chain cannot be observed directly. We can observe only their effect through the functions b_j.

In speech recognition, the observations are sequences of vectors of parameters such as LPC coefficients, PARCOR coefficients or MFCC coefficients. There exist other parametric representations of speech, but in this chapter, we will use the MFCC parameters because we have seen in chapter 5 that they are near optimum for discrimination of patterns. The type of Markov chains used in speech recognition are not ergodic, but "left to right" as shown in Figure 6-7.

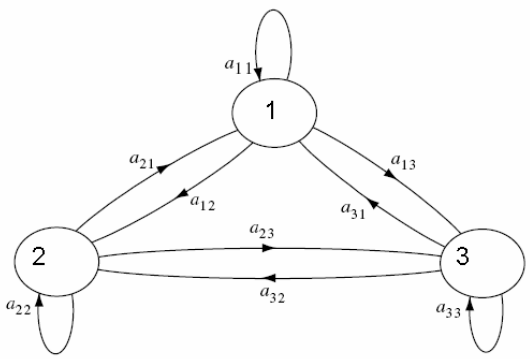


Figure 6-6 Ergodic HMM Structure

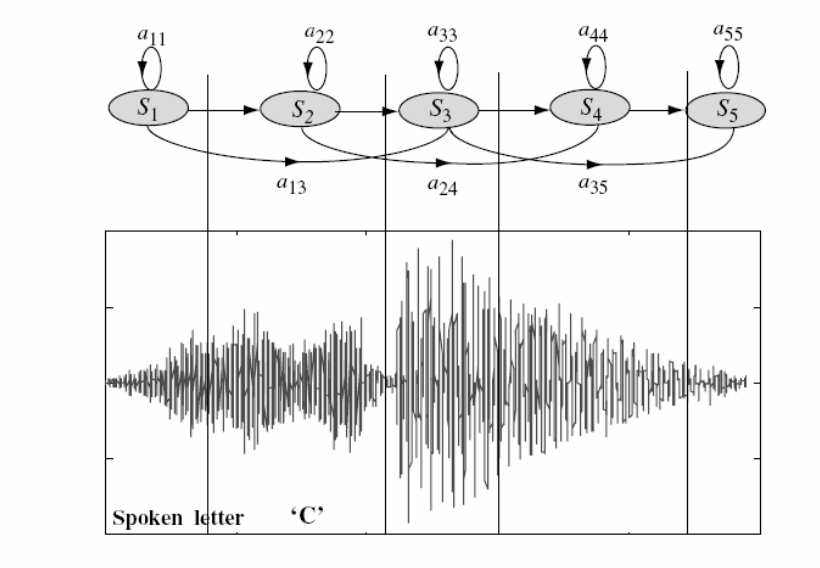


Figure 6-7 Five State Left to Right HMM Speech Model [92]

In the left to right Markov chain, the transition matrix **A** is upper triangular, and is usually assumed to be banded. The transition matrix corresponding to the state transition diagram shown in Figure 6-7 is:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} \\ 0 & 0 & 0 & a_{44} & a_{45} \\ 0 & 0 & 0 & 0 & a_{55} \end{pmatrix} \tag{6.8}$$

A being a stochastic matrix has the property that $a_{NN} = 1$. The spoken letter "C" corresponds to the two phones /ss/ and /ii/ cascaded. The self loops ($a_{ii} \neq 0$) indicate that the corresponding speech segment can have an arbitrary long duration. The first off diagonal ($a_{k,k+1}$) indicate a normal transition while the second off diagonal ($a_{k,k+2}$) indicate that the next state can be skipped. We see clearly that we can model very accurately the time variations of the speech waveform. When we use a right to left topology, we have to start at the first state. This implies that the initial state probabilities form the following vector:

$$\boldsymbol{\pi} = \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix} \tag{6.9}$$

So, when we are pronouncing "C", the associated SFSA will go from one state to another until we arrive to the last state, which is an acceptor state. This means that in order to recognize a particular speech segment of length $T + 1$ frames, we have to find the particular state sequence $\{q_0, q_1, \ldots, q_T\}$ it has undertaken given the sequence of

measurements $\{\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_T\}$. If we use the concept of MAP defined by (6.1), this means that we must find the path in the state trellis that maximizes $P[q_0, q_1, \ldots, q_T|\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T]$. An exhaustive search through all possible paths will be too costly since the total number of possible paths depends exponentially on the number of states N and on the number of measurements $T+1$. A solution to this problem is provided by the Viterbi Algorithm [95, 33, 96]. The Viterbi algorithm is essentially a dynamic programming method [10] and is equivalent to the optimum methods for finding the shortest path in graph theory. It has been developed originally for decoding short constraint length convolutional codes [95]. Using the Bayes theorem, maximizing the probability of the state sequence given the observation sequence is equivalent to maximizing the following product:

$$P\left[\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T \,|q_0, q_1, \ldots, q_T\right] P\left[q_0, q_1, \ldots, q_T\right] \tag{6.10}$$

(6.10) can be rephrased as:

$$\prod_{k=0}^{T} P\left[\mathbf{x}_k \,|q_k\right] \prod_{k=0}^{T} P\left[q_k \,|q_{k-1}\right] \tag{6.11}$$

if we set $P\left[q_0 \,|q_{-1}\right] = P\left[q_0\right]$. At that time, we iteratively solve the maximization problem. We define by $\delta_t(i)$ the maximum value of the metric (6.10) of the path arriving at state i at frame index t. Then, using the dynamic programming principle that any sub-path of an optimum path is itself optimum, using (6.11), we can state the following recurrence rule:

$$\delta_{t+1}(j) = b_j(\mathbf{x}_{t+1}) \max_{1 \le i \le N} \left[\delta_t(i) a_{ij}\right] \tag{6.12}$$

since $b_j(\mathbf{x}_{t+1}) = P\left[\mathbf{x}_{t+1} \,|q_{t+1} = j\right]$ and $a_{ij} = P\left[q_{t+1} = j \,|q_t = i\right]$. Because (6.12) is a product of numbers with modulus less than one (probabilities), the above quantity is numerically unstable. It is better to use the logarithm of the above quantity as a path metric and replace the multiplications by additions.

$$\tilde{\delta}_{t+1}(j) = \tilde{b}_j(\mathbf{x}_{t+1}) + \max_{1 \le i \le N} \left[\tilde{\delta}_t(i) + \tilde{a}_{ij}\right] \tag{6.13}$$

where $\tilde{\delta}_t(i) = \log\left[\delta_t(i)\right]$, $\tilde{b}_i(\mathbf{x}_t) = \log\left[b_i(\mathbf{x}_t)\right]$ and $\tilde{a}_{ij} = \log\left[a_{ij}\right]$.

In continuous speech recognition, we model each phone by a tri-state HMM and the phone is recognized if it has the highest end path metric (6.13). The word is then recognized by the use of a dictionary of phonemic transcriptions of the given vocabulary.

A major problem in all recognition systems is the evaluation of the different parameters that form the model. In our case, we must estimate the different transition probabilities a_{ij} and evaluate the probability density functions $b_j(\mathbf{x})$. This is achieved by "training" the recognizer. In the case of HMM, a powerful training method, the Baum-Welch algorithm [8, 98] has been developed. A description of the algorithm is provided in ref. [16, 72, 73, 76, 98].

6.4 Benchmark System [76]

As a benchmark continuous speech recognition system, we are going to use a toolkit developed at the Cambridge University, the Hidden Markov model Tool Kit (HTK). It is a set of files that can be downloaded freely from the Cambridge University site http://htk.eng.cam.ac.uk/. The toolkit can be used for general purpose modelling of time series using HMM's. However, HTK is primarily designed for building and training speech recognizers.

The system designed using HTK is a continuous speech recognition for colloquial Algerian Arabic numbers from one to 999 999 999 [76]. When somebody wants to develop an HMM based recognition system in American English, he can use a very powerful corpus of speech data, TIMIT Acoustic-Phonetic Continuous Speech Corpus [30]. It consists of a database of manually time aligned speech data from 630 speakers using eight dialects of American English. However, such system is not available for Algerian Arabic. So, data was collected from 37 department students forming a database of 11230 sentences pronounced by 17 male students and 20 female students.

The speech signal is acquired with a sampling rate of 16 bits along with 16 bits per samples using built in sound cards of PC computers. This data has been manually segmented into phones using a freely available software, Speech Analyzer ver.1.5 from Summer Institute of Linguistics, copyright © 1996-2002, Acoustic Analysis Project, IAARS-CCS, Waxhaw, N.C., USA.

A task grammar has been developed to enhance the recognition using syntactic rules. The language to be recognized is the central Algerian dialect. The task grammar along with pronunciation rules is much more complex than classical or standard Arabic language due to the lack of a unique rule for pronouncing the different words. For example, the digit two is either "zoudj" or "tnin". This second form can have different pronunciations such as /tt/ /nn/ /ii/ /nn/ or /st/ /nn/ /ii/ /nn/.

Figure 6-8 shows a graphical representation of the top level of the task grammar used to define our recognition scheme. Any spoken number between one and 999 999 999 will always obey this grammar. The block named $units defines the structure of numbers between one and 999, the block $thousands defines the structure of numbers between one thousand and 999 thousands and finally, the block $millions defines the structure of numbers between one million and 999 millions. The word composed of the single phoneme "u" (/oo/) links the different blocks together and has the meaning of the English word "and".

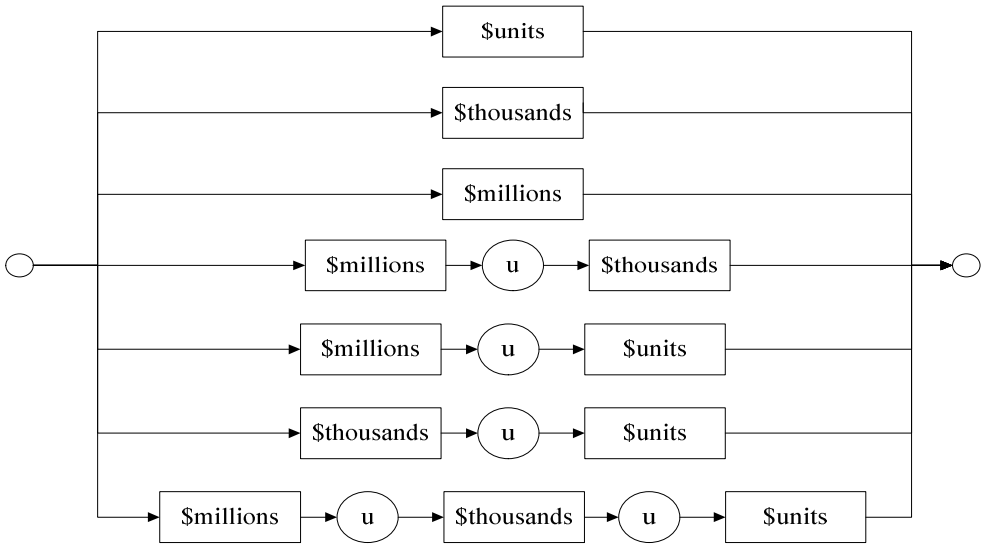


Figure 6-8 Overall Task Grammar [76]

Each block in Figure 6-8 can be developed as shown in Figure 6-9 and Figure 6-10. However, describing the whole task grammar graphically is too lengthy. We are going to use HTK word building language which is based on the extended Backus-Naur Form (EBNF) description language [81].

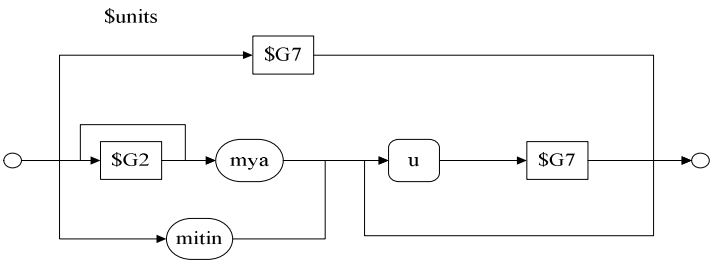


Figure 6-9 Development of the $units Block

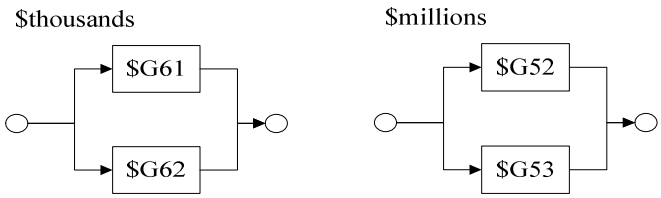


Figure 6-10 Development of the $thousands and $millions blocks

The definition of the task grammar is given in a bottom up manner, starting from basic sub-blocks all the way to the final definition of the complete grammar. Each block contains words or parts of words that share some common acoustic properties. For example, we can connect the phoneme "a" (/aa ita/) to the sub-block $G0 to produce the words: "reb3a" (/rr/ /aa/ /bb/ /3a/ /aa ita/), "khemsa" (/Kh/ /aa/ /mm/ /ss/ /aa ita/), "setta" (/ss/ /aa/ /tt/ /aa ita/), "seb3a" (/ss/ /aa/ /bb/ /3a/ /aa ita/), "tes3" (/tt/ /aa/ /ss/ /3a/ /aa ita/). These words are the representations of the numbers 4, 5, 6, 7, 9 respectively.

Definition of the task grammar:

```
$G0   ::= reb3 | khems | sett | seb3 | tes3;
$G1   ::= thlath | thmany | $G0;
$G11 ::= 3eshr | $G1;
$G2   ::= thelth |themn|$G0;
$G22 ::= $G2|3eshr;
$G23 ::= zoudj|$G2|3eshr;
$G3   ::= hdash|thnash|(thlet|rbe3|khmes|seT|sbe3|thmen|tse3) Tash;
$G40 ::= wahed | thnin | $G1 a;
$G4   ::= [$G40 u] $G1 a;
$G50 ::= $G3 en | $G4;
$G51 ::= u [$G3 en | $G4];
$G52 ::= [[$G2] mya (t | $G51) | mitin [$G51] | $G50] melyun;
$G53 ::= [(([$G2] mya) | mitin) u ] $G23 mlayen;
$millions ::= $G52 | $G53;
$G6   ::= $G22 alaf | alfin;
$G61 ::= [[$G2] mya (t | $G51) | mitin [$G51] | $G50] alef;
$G62 ::= [(([$G2] mya) | mitin) u] $G6;
$thousands ::= $G61 | $G62;
$G7  ::= wahed | zoudj | $G11 a | $G3 | $G4;
$units ::= $G7 | ([$G2] mya | mitin) [u $G7];
(SENT-START ($units | $thousands | $millions | $millions u $thousands | $millions u
$units | $thousands u $units | $millions u $thousands u $units) SENT-END)
```

The last line give the definition of the overall grammar starting by a silence represented by the word SENT-START and ending by another silence SENT-END.

HTK requires a dictionary containing all the words to be recognized listed in alphabetical order. The dictionary contains the words with their phoneme transcription.

The system implemented does not use the phonemes listed in appendix A, but a compatible transcription of these phonemes in a form that can be accepted by HTK. The sentences used to build the database have been generated by HTK and then pronounced by the selected students. All the sentences are then syntactically correct according to the previously defined task grammar. These sentences have been used to train the system.

The recognition and the training of the system based on HTK used the same acquisition system. The raw speech waveform is sampled at 16 kHz. It is then windowed using a 256 samples Hamming window that will be translated by 100 samples for each frame as shown in Figure 4-15. We have seen in chapter 5 that the MFCC parameters form the "best" set of parameters. Therefore, we select 13 MFCC parameters using 26 channels. We add the ΔMFCC and the ΔΔMFCC to have a better representation of transient phonemes. This means that we have a feature space of size 39.

The system to be designed is continuous speech recognition. Therefore, we model every phone (monophone) by a tri-state HMM. The data is manually labeled and we use the training system built in HTK. The training is based on the Baum-Welch algorithm. The generation probability density functions $b_j(\mathbf{x})$ are multivariate Gaussian densities. Because of the high dimensionality, the covariance matrices are assumed diagonal. This eliminates the risk of singular covariance matrix due to a lack of data in the training set. This diagonality of the estimated covariance implies that we have assumed that the components of the observations are independent. This assumption is not always true but reduces the computation of the covariance matrix to the computation of 39 variances and the computation of the inverse covariance to the inverse of the diagonal elements. It also happens that this assumption is not detrimental to the recognition rate of the final system.

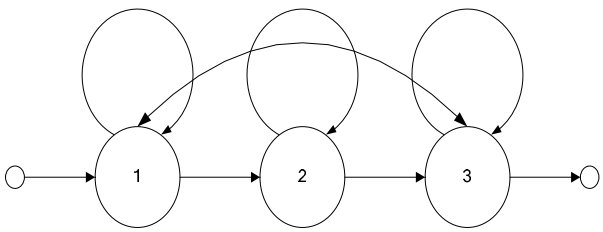


Figure 6-11 Silence Model

Figure 6-11 shows a tri-state model for silence. We do not use a classical left to right HMM. We can remark a reverse path from right to left. The backward skip impedes the model to transit to the following word in case of impulsive noise. A short

pause single state HMM has also been added. This short pause shares the center state of the silence model. These models have also been trained. An interesting feature of HTK is the possibility to create triphone models from the previously defined monophones. This is achieved by tying the models together. Triphone modeling allows the recognition system to take into account coarticulation, which is the influence of one phoneme on the next one.

Benchmark system results [76]:

The system was tested with 200 test utterances (test sentences). The test utterances were taken at random from the collected database. 194 were recognized correctly and there were five substitution errors. This gives a sentence recognition rate of 96.90%. The substitution errors affect only some words in the sentences. At the word level, from a total of 855 words, 853 were recognized correctly. The recognition rate is then 99.0%. There was one deletion error, no substitution error and one insertion error. We see clearly that the designed recognizer is very accurate.

6.5 Sequential Decoding Applied to Speech Recognition [20]

In the previous part, we have described a powerful system designed around HMM's. One drawback of the HMM systems is the large amount of data needed for training. The Baum-Welch algorithm will compute the transition probabilities along with the production densities. We decided to design a continuous speech recognition system that requires much less data for training but that keeps the flexibility of the HMM modeling. Our approach is based on the model of Figure 1-3, which is repeated below for convenience.

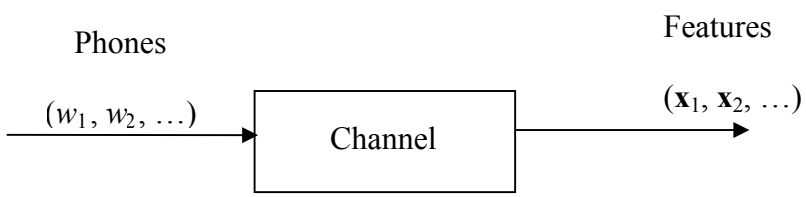


Figure 6-12 Speech Recognition Model

When we want to produce a word, the human brain will translate the word into a sequence of phoneme. Then every phoneme is translated (encoded) into a sequence of realizations (phones). The output of the channel is going to be a sequence of measurements (MFCC coefficients in our case).

In general, the channel is a non-stationary memory type of channel. However, considered as such, it will lead to an almost intractable problem. This representation can be simplified by modeling the channel as a stationary, discrete input, continuous output, memoryless channel. Therefore, if we consider the conditional probabilities of sequences of output vectors given a particular sequence of phones, we have:

$$\begin{aligned} &\Pr[\mathbf{X}|\mathbf{W}] = \prod \Pr[\mathbf{x}_i | w_i] \\ &\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots) \\ &\mathbf{W} = (w_1, w_2, \ldots) \end{aligned} \tag{6.14}$$

It has been shown in many previous works [73, 74] that such assumption is not detrimental to the final result, which is a good recognition rate.

The assumed encoder is going to be modeled by a finite state automaton. When a word is pronounced, a path is traced in the corresponding trellis. The speech recognition task will be the identification of the most likely path.

6.5.1 Speech Production Model

The models used for representing the previously described system are deterministic finite state automata. FSA can be shown to be equivalent to HMM's if all the transition probabilities are equal. So, in our case, we use the following definition:

The phone production model is completely defined by the quadruple (S, Γ, f, g) where:

1. $S = \{1, 2, ..., N\}$ is a finite non empty set of states. A specific state visited at frame index t is denoted q_t.
2. $\Gamma = \{z_1, z_2, ..., z_M\}$ is a finite non empty set of output symbols. The symbols are the phone transcriptions.
3. $f : S \rightarrow S$ is a state transition function which take the previous state and produces the present state $q_t = f(q_{t-1})$.
4. $g : S \times S \rightarrow \Gamma$ is a production function which takes the previous and the present state and returns an output symbol.

As defined, the speech production model is going to be a Mealy type of FSA. Our model is going to be based essentially on phonemes characteristics. We have already classified phonemes according to their durations and parameter variations in two main classes:

- Sustainable or non-transient phoneme that can have a quite long duration (such as vowels).
- Non-Sustainable or transient phonemes, which have a very short duration but a quite large parameter variations (such as plosives).

In this work, we model speech an ordered sequence of the previously classified speech units. In our model even the natural silence is modelled the same way. Our approach is intermediate between the one taking by Scagliola and Marni [80] and the mostly used speech modelling method in speech recognition which h is HMMs method. Hence each word will be represented by a finite state automata constructed following the given word sound units (phonemes) models.

Actually all of the words, which form the vocabulary, to be recognized will be represented by a single finite state automaton.

In this model, sustainable phonemes are supposed to have an infinite duration while plosives (transients) have a finite duration. We have seen in chapter 2 that a transient can have at most duration of 100 ms. In this implementation, we have used a sampling rate of 16 kHz. The frames are translated by 100 samples every time and each frame is 256 samples (we are using overlapping frames as in the previous system).This means that a transient cannot have a duration that exceeds 10 frames.

So, a sustainable is represented by a single state with an input from the previous phoneme, a self loop and an output to the coming phoneme (see Figure 6-13) while a transient (plosive) is a ten states automaton where we have possibility of transition from the initial state to any other state in the automaton, transitions from all succeeding states up to the final state and there is no self loop for any state (see Figure 6-14 where only 5 states are represented).

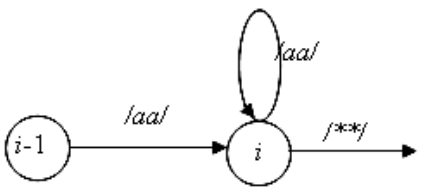


Figure 6-13 State Diagram for the Vowel /aa/

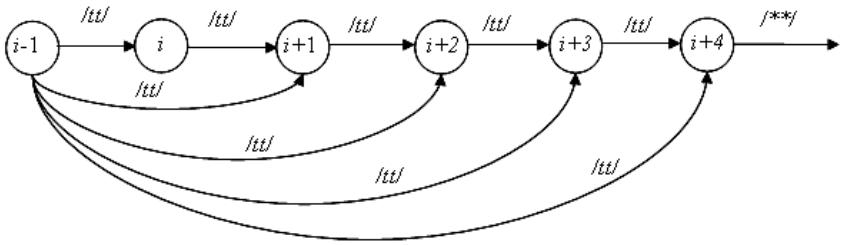


Figure 6-14 State Diagram for the Plosive /tt/

The symbol /**/ represents any other phonemes but one that is in our state diagram. When a word is pronounced, it will define a path in the trellis corresponding to the considered vocabulary. Using the above representation, an isolated word single locutor system has been developed. It provides a recognition rate of nearly 100% for a limited vocabulary [1]. Using the same model, a single locutor word-spotting algorithm has also been developed [60]. It also provides the same recognition rate.

Table 6-2 shows the definition of the first five words in our vocabulary ("wahed", "zoudj", "tleta", "arb3a", "khemsa") with only 5 state for transients as illustration. In the training part, we have developed a software program that updates automatically the automaton every time a new word is added.

Present State	Next State	Path Label
0	1	/wa/
1	1	/wa/
1	2	/ha/
2	2	/ha/
2	3	/ad/
2	4	/ad/
2	5	/ad/
2	6	/ad/
2	7	/ad/
3	4	/ad/
4	5	/ad/
5	6	/ad/
6	7	/ad/
0	8	/zz/
8	8	/zz/
8	9	/oo/
9	9	/oo/
9	10	/jj/
9	11	/jj/
9	12	/jj/
9	13	/jj/
9	14	/jj/
10	11	/jj/
11	12	/jj/
12	13	/jj/
13	14	/jj/
0	15	/tt/

0	16	/tt/
0	17	/tt/
0	18	/tt/
0	19	/tt/
15	16	/tt/
16	17	/tt/
17	18	/tt/
18	19	/tt/
19	20	/ll/
20	20	/ll/
20	21	/aa ita/
21	21	/aa ita/
21	22	/tt/
21	23	/tt/
21	24	/tt/
21	25	/tt/
21	26	/tt/
22	23	/tt/
23	24	/tt/
24	25	/tt/
25	26	/tt/
26	27	/aa ita/
27	27	/aa ita/
0	28	/aa/
28	28	/aa/
28	29	/rr/
29	29	/rr/
29	30	/bb/
29	31	/bb/
29	32	/bb/
29	33	/bb/
29	34	/bb/
30	31	/bb/
31	32	/bb/
32	33	/bb/
33	34	/bb/
34	35	/ʒa/
35	35	/ʒa/
0	36	/kh/
36	36	/kh/
36	37	/mm/
37	37	/mm/
37	38	/ss/
38	38	/ss/
38	39	/aa ita/
39	39	/aa ita/

Table 6-2 State Transition Table for the First 5 Words

Every phone w_i is a path label and defines a conditional probability density function $f_{\mathbf{X}|w_i}\left(\mathbf{x} \mid w_i\right)$. These phones are trained individually. During the development of the system, the different phonemes have been extracted manually from the training data. The PDF's are multivariate Gaussian. Therefore, they are completely defined by their mean vector and covariance matrix.

6.5.2 The Stack Algorithm

When the vocabulary is limited, the corresponding state diagram is quite small (small number of states). The recognition can be performed using the Viterbi algorithm. When the size of the vocabulary increases, the number of states becomes prohibitively large for the Viterbi algorithm to perform properly. In this case, we suggest using the "*stack algorithm*" [48, 105]. The stack algorithm is a member of the family of sequential algorithms [102], which are commonly used to decode long constraint length convolutional codes.

Sequential algorithm are algorithms that make the hypothesis that a given path is the optimum path and they will extend it until the end of the path or until the hypothesis is proved wrong. At that time, there should be a strategy for going back, select a new path and repeat the process. If it is possible to store all possible hypotheses, the sequential algorithm will be optimal (just as the Viterbi algorithm). However, it is generally impossible to do so. At that time, the correct hypothesis can be dropped out of the selected data structure and it will be impossible to decode correctly the given word.

The stack algorithm follows the subsequent steps:

1. The stack algorithm maintains in a stack partial hypotheses of all paths sorted in descending order of likelihood (path metric). A partial hypothesis accounts for an initial frame of the input speech.
2. It pops out the best path of the stack (top of the stack) and expands it by all possible state extensions predefined in the state diagram (automaton). It evaluates the resulting partial hypothesis with respect to the input speech and re-inserts them in the sorted stack.
3. If the hypothesis is completed, it is output. Otherwise, the algorithm reiterates the second step until a complete hypothesis is found.

We say that the hypothesis is complete when the path exits the last state of a given word. At that time, we can declare that the word has been recognized. The data structure used to hold the partial hypotheses must have a finite size. This means that the paths with very small metric (highly unlikely) will be dropped from the stack. In adjusting the size of the stack, we must make sure that the probability of dropping the

correct hypothesis is very small. For the implementation of the stack algorithm, we need a path metric that remains constant or near zero we are following a correct path. Alternatively, the path metric should drop very rapidly when we are following an incorrect path. The proposed path metric is:

$$L = \sum_{path\ k} L_i(\mathbf{x}) \tag{6.15}$$

where $L_i(\mathbf{x}) = p - d_i^2$ and $d_i^2 = (\mathbf{x} - \mathbf{m_i})^T \boldsymbol{\Sigma}_\mathbf{i}^{-1} (\mathbf{x} - \mathbf{m_i})$ is the Mahalanobis distance between the received utterance $\mathbf{x}$ and the local path reference mean $\mathbf{m}_i$. p is the dimensionality of the feature space and $\boldsymbol{\Sigma}_\mathbf{i}$ is the i[th] class covariance matrix. The above path metric has practically the same properties as the Fano [27] metric used in sequential decoding of convolutional codes. We can show that $L_i(\mathbf{x})$ remains small on average if we are on the correct path and has a large negative value on an incorrect path (on average). In order to do so, we must compute the average of the Mahalanobis distance on a given branch of a path under analysis.

If the samples are drawn from population i and classified as belonging to population j, then we can compute the following average:

$$E[d_{ij}^2] = \int (\mathbf{x} - \mathbf{m}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \mathbf{m}_j) N(\mathbf{x}, \mathbf{m}_i, \boldsymbol{\Sigma}_i) d\mathbf{x} \tag{6.16}$$

where $N(\mathbf{x}, \mathbf{m}_i, \boldsymbol{\Sigma}_i)$ is the normal density with mean $\mathbf{m}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$. Let $\mathbf{m}_{ij} = \mathbf{m}_j - \mathbf{m}_i$, it is easy to show that:

$$E[d_{ij}^2] = \int (\mathbf{x} - \mathbf{m}_{ij})^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \mathbf{m}_{ij}) N(\mathbf{x}, 0, \boldsymbol{\Sigma}_i) d\mathbf{x} \tag{6.17}$$

Using the eigen decomposition of the matrix $\boldsymbol{\Sigma}_i$, i.e. , let us consider the orthonormal matrix $\mathbf{V}_i$ composed of the normalized eigenvectors of the matrix $\boldsymbol{\Sigma}_I$ and the diagonal matrix $\boldsymbol{\Lambda}_I$ composed of the eigenvalues of $\boldsymbol{\Sigma}_i$:

$$\boldsymbol{\Sigma}_i \mathbf{V}_i = \mathbf{V}_i \boldsymbol{\Lambda}_i \tag{6.18}$$

By making the transformation: $\mathbf{x} = \mathbf{V}_i \mathbf{y}$, we obtain:

$$E[d_{ij}^2] = \int (\mathbf{y} - \mathbf{V}_i^T \mathbf{m}_{ij})^T \mathbf{V}_i^T \boldsymbol{\Sigma}_j^{-1} \mathbf{V}_i (\mathbf{y} - \mathbf{V}_i^T \mathbf{m}_{ij}) N(\mathbf{y}, 0, \boldsymbol{\Lambda}_i) d\mathbf{y} \tag{6.19}$$

Now, we have two possibilities. Either we are on a correct branch, and $i = j$, or we are on an incorrect branch of the path, and in this case $i \neq j$. In the first case, i.e. the correct branch, we obtain:

$$E[d_{ii}^2] = \int \mathbf{y}^T \mathbf{V}_i^T \boldsymbol{\Sigma}_j^{-1} \mathbf{V}_i \mathbf{y} N(\mathbf{y}, 0, \boldsymbol{\Lambda}_i) d\mathbf{y} \tag{6.20}$$

In this case, d_{ii}^2 is just a sum of squares of zero mean, unit variance, uncorrelated Gaussian random variables. Therefore, it is a chi-square random variable with p degrees of freedom and the above mean has the value of the dimension of the observation space p.

When we are on an incorrect path, i.e. $i \neq j$, the above mean is much more difficult to compute. However, a great simplification can be made if we assume that $\mathbf{\Sigma}_i = \mathbf{\Sigma}_j = \mathbf{\Sigma}$. In this case:

$$E[d_{ij}^2] = \int (\mathbf{y} - \mathbf{V}^T\mathbf{m}_{ij})^T \mathbf{V}^T \mathbf{\Sigma}^{-1} \mathbf{V} (\mathbf{y} - \mathbf{V}^T\mathbf{m}_{ij}) N(\mathbf{y}, 0, \mathbf{\Lambda}) d\mathbf{y} \qquad (6.21)$$

or

$$E[d_{ij}^2] = \int (\mathbf{x} - \mathbf{V}^T\mathbf{m}_{ij})^T \mathbf{\Lambda}^{-1} (\mathbf{x} - \mathbf{V}^T\mathbf{m}_{ij}) N(\mathbf{x}, 0, \mathbf{\Lambda}) d\mathbf{x} \qquad (6.22)$$

We assume that the covariance matrix is positive definite. We can then make the following simplification:

$$\mathbf{x}^T \mathbf{\Lambda}^{-1} \mathbf{x} = \mathbf{x}^T \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{x} = \mathbf{z}^T \mathbf{z} \qquad (6.23)$$

Meaning that we make the following change of variables:

$$\mathbf{z} = \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{x} \qquad (6.24)$$

and we obtain:

$$E[d_{ij}^2] = \int (\mathbf{x} - \mathbf{V}^T\mathbf{m}_{ij})^T \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{\Lambda}^{-\frac{1}{2}} (\mathbf{x} - \mathbf{V}^T\mathbf{m}_{ij}) \; N(\mathbf{x}, 0, \mathbf{\Lambda}) d\mathbf{x}$$

$$E[d_{ij}^2] = \int (\mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{x} - \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{V}^T\mathbf{m}_{ij})^T (\mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{x} - \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{V}^T\mathbf{m}_{ij}) N(\mathbf{x}, 0, \mathbf{\Lambda}) d\mathbf{x} \qquad (6.25)$$

and finally:

$$E[d_{ij}^2] = \int (\mathbf{z} - \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{V}^T\mathbf{m}_{ij})^{\mathbf{t}} (\mathbf{z} - \mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{V}^T\mathbf{m}_{ij}) N(\mathbf{z}, \mathbf{0}, \mathbf{I}) d\mathbf{z} \qquad (6.26)$$

This implies that d_{ij}^2 is a non-central chi-square random variable with p degrees of freedom with a mean for each of the variables being:

$$\mu_k = (\mathbf{\Lambda}^{-\frac{1}{2}} \mathbf{V}^T\mathbf{m}_{ij})_k \qquad (6.27)$$

Therefore, the above mean is:

$$E[d_{ij}^2] = p + 2\mathcal{N} \qquad (6.28)$$

where

$$\mathcal{N} = \frac{1}{2} \sum_{k=1}^{N} \mu_k^2 \qquad (6.29)$$

so $$E[d_{ij}^2] = p + \sum_{k=1}^{N}\left(\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{V}^T\mathbf{m}_{ij}\right)_k^2 \tag{6.30}$$

The above expression can be further simplified by noticing that:

$$\sum_{k=1}^{p} z_k^2 = \|\mathbf{z}\|^2 = \mathbf{z}^T\mathbf{z} \tag{6.31}$$

So:

$$\begin{aligned} E[d_{ij}^2] &= p + (\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{V}^T\mathbf{m}_{ij})^T\ (\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{V}^T\mathbf{m}_{ij}) \\ &= p + \mathbf{m}_{ij}^T\mathbf{V}\boldsymbol{\Lambda}^{-1}\mathbf{V}^T\mathbf{m}_{ij} \\ &= p + \mathbf{m}_{ij}^T\boldsymbol{\Sigma}^{-1}\mathbf{m}_{ij} \end{aligned} \tag{6.32}$$

We can remark that if we use p- d^2 as local path metric, then on a correct path, it has a high probability to be equal to zero. However, on an incorrect path, the average of the above path metric is the negative number $-\mathbf{m}_{ij}^T\boldsymbol{\Sigma}^{-1}\mathbf{m}_{ij}$. So, it is proved that p - d^2 is a good candidate for a path metric.

When we are on a wrong path, the local path metric is going to decrease. Averaging over all classes, the following result is obtained. Consider that the phones are clustered in M different equiprobable classes. The average decrease of the local path metric is proportional to following summation:

$$\begin{aligned} S &= \sum_{i=1}^{M}\sum_{j=1}^{M}\mathbf{m}_{ij}^T\boldsymbol{\Sigma}^{-1}\mathbf{m}_{ij} \\ &= \sum_{i=1}^{M}\sum_{j=1}^{M} tr[\mathbf{m}_{ij}^T\boldsymbol{\Sigma}^{-1}\mathbf{m}_{ij}] \\ &= \sum_{i=1}^{M}\sum_{j=1}^{M} tr[\boldsymbol{\Sigma}^{-1}\mathbf{m}_{ij}\mathbf{m}_{ij}^T] \\ &= tr\left[\boldsymbol{\Sigma}^{-1}\sum_{i=1}^{M}\sum_{j=1}^{M}\mathbf{m}_{ij}\mathbf{m}_{ij}^T\right] \end{aligned} \tag{6.33}$$

Introducing now the overall average:

$$\mathbf{m}_0 = \frac{1}{M}\sum_{i=1}^{M}\mathbf{m}_i \tag{6.34}$$

We can decompose the product $\mathbf{m}_{ij}\mathbf{m}_{ij}^T$ as:

$$\mathbf{m}_{ij}\mathbf{m}_{ij}^T = (\mathbf{m}_i-\mathbf{m}_0)(\mathbf{m}_i-\mathbf{m}_0)^T + (\mathbf{m}_j-\mathbf{m}_0)(\mathbf{m}_j-\mathbf{m}_0)^T - (\mathbf{m}_i-\mathbf{m}_0)(\mathbf{m}_j-\mathbf{m}_0)^T - (\mathbf{m}_j-\mathbf{m}_0)(\mathbf{m}_i-\mathbf{m}_0)^T$$

and the double summation becomes:

$$\begin{aligned}\sum_{i=1}^{M}\sum_{j=1}^{M}\mathbf{m}_{ij}\mathbf{m}_{ij}^{T} &= M\sum_{i=1}^{M}(\mathbf{m}_i-\mathbf{m}_0)(\mathbf{m}_i-\mathbf{m}_0)^T \\ &+ M\sum_{j=1}^{M}(\mathbf{m}_j-\mathbf{m}_0)(\mathbf{m}_j-\mathbf{m}_0)^T \\ &+ \sum_{i=1}^{M}(\mathbf{m}_i-\mathbf{m}_0)\sum_{j=1}^{M}(\mathbf{m}_j-\mathbf{m}_0)^T \\ &+ \sum_{j=1}^{M}(\mathbf{m}_j-\mathbf{m}_0)\sum_{i=1}^{M}(\mathbf{m}_i-\mathbf{m}_0)^T\end{aligned} \tag{6.35}$$

The last two terms are zero, so, finally, if we introduce the between mean covariance (scatter) matrix **B**:

$$\mathbf{B} = \frac{1}{M}\sum_{i=1}^{M}(\mathbf{m}_i-\mathbf{m}_0)(\mathbf{m}_i-\mathbf{m}_0)^T \tag{6.36}$$

The summation *S* becomes:

$$S = 2M^2 tr[\boldsymbol{\Sigma}^{-1}\mathbf{B}] \tag{6.37}$$

Finally, we have shown that minimizing the average decrease of the local path metric $p\text{-}d^2$ over all possible classes amounts to maximizing $tr[\boldsymbol{\Sigma}^{-1}\mathbf{B}]$. Therefore, the above path metric is a good candidate for a path metric if the feature set is selected such that the above quantity is maximum. The value *S* given by equation (6.37) is the same as the *J* criterion given by equation (5.30) in chapter 5. In our analysis, we have assumed that the features are Gaussian. We have seen in chapter 5 that the MFCC parameters are practically Gaussian and it is the set that maximizes the *J* criterion.

6.5.3 Implementation of the Algorithm

In this implementation, we are going to use a sampling rate of 16 kHz. The data is windowed using the same Hamming window of 256 samples as the previous system and this window is translated by 100 samples for each frame. The set of parameters used to represent every frame is the set of 13 MFCC parameters. We did not use the derivative and the acceleration parameters in our system. The language to be recognized is the same set of Algerian numbers as the previous system. Every phone from Table A-1 has been manually trained and we have used a full (not diagonal) covariance matrix for each class.

We have used a stack algorithm where we extend the path with the highest metric. The different words that compose the vocabulary are identified by the beginning and ending state. So, a word is recognized if the winning path finishes by a branch that

comes out from the ending state of a word in the vocabulary. In order to limit the size of the stack, we are going to use a stack with a size of about twice and a half the size of the vocabulary. The drawback of the limitation of the stack size is the fact that if the correct path happens to be eliminated (being at the bottom of the stack), then it is impossible for the system to recognize the word. However, it is shown below that such event is very improbable.

We have used also the following property for rejecting a very improbable path. If we are on a correct path, then d^2 is a chi-square random variable with p degrees of freedom (p = 13 in our case). At that time, $\text{Probability}\left[d^2 \geq 29.82\right] = 0.005$ (0.5%). So, if we are on the correct path, then the local path metric $L_i(\mathbf{x}) = p - d_i^2$ is larger that -16.82 with a probability of 99.5%. So, if after K steps, we find that a path has a global metric L smaller than -16.82K, then there is a high probability for that path to be incorrect. We have used the algorithm for colloquial Algerian Arabic numerals. For the word "thmania", we found that the winning path oscillates between 35 and -45 for a path length of about 130 sections. Incorrect paths for this particular case all go down at a rate faster than the above mentioned one ($-16.82K$). One rule for pruning the tree search is: if a path has a global metric of about -168.2 after 10 steps, it has a very small probability of being part of the correct one since each one of its branches has a probability of 0.5% of belonging to the correct one. This particular value of d^2 is also used in the training program for generating a new class.

An advantage of having a global metric that averages to zero is the fact that we can start a new recognition without having to reset the previous search. The wrong path will drop down through the stack and the correct one will start from a value that is close to zero. The extension will come from the grammar that is imbedded inside the finite state automaton. We have started by a word spotting system.

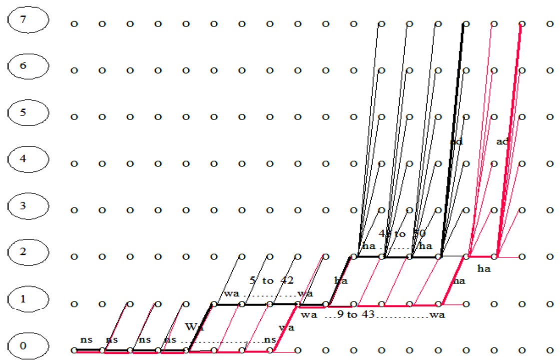


Figure 6-15 Winning Path for theWord "wahed"

Figure 6-15 shows the winning path for two different speakers. The black path is the winning path for the first locutor (male speaker), while the pink one corresponds to the second one (female speaker). We distinguish between the male and the female speaker because the actual probability density function for almost all the different classes (*phones*) appears to be bimodal. Therefore, we replace every bimodal one by two unimodal distributions with the same label. The light lines in Figure 6-15 show some paths that were rejected by the algorithm. So, the male speaker has pronounced the phone /wa/ from time frame 4 all the way to time frame 42 and then /ha/ from time frame 43 to 50 and finally /ad/ for the last time frame. (A frame consists of 256 samples of speech taken at rate of 16 kHz translated by 100 samples). For the spotting of words in a continuous flow of words, we have added a state "*not in the vocabulary*" in the automaton.

When we tested the word-spotting algorithm with the training data, we obtained a 100% recognition rate. However, when we used the same 200 utterances as in the previous system, we obtained 182 correct utterances; this provides a recognition rate of 91% at the sentence level. At the word level, from a total of 855 words, 843 were recognized correctly. The recognition rate is then 98.0%.

In order to obtain better results, we have embedded the finite state automaton representing the vocabulary inside the finite state automaton representing the task grammar defined in section 6.4. We have also added some pre-processing as described in chapter 3 (when the acquired speech is clipped) and denoising using time varying Wiener filters as shown in chapter 4. We repeated the same tests. We obtained better results. The results are of the same order as the ones obtained using HTK. For the 200 utterances, 193 were correct. The recognition rate has increased to 96.5% at the sentence level. At the word level, we obtained six errors so the recognition rate in this case is 99%. We have also remarked that most of the errors occurred for words that contain transient phones. We think that the addition of ΔMFCC will provide a better recognition rate.

The stack algorithm along with the adopted path metric has provided very interesting results despite the fact that we did not use the derivative and acceleration parameters. The main difficulty comes from the training. In the next chapter, we are going to show an automatic way of segmenting speech in order to extract training parameters.

Chapter 7

Automatic Training

In the previous chapter, we had to train the speech recognizer. This means that we had to acquire data from spoken sentences. Since the recognition is phoneme based, speech had to be manually segmented in these basic units. And since we are using supervised parametric learning [35, 88], the means and covariance matrices of the different classes are computed. In order to have satisfactory statistical results, a large amount of data is needed. Manual segmentation is a tedious process and it is quite error prone. In this chapter, we are going to show that we can automatically segment speech and update pre-computed parameters.

7.1 Finite State Automaton Update

The first step in the training is the introduction of the word state sequence in the total vocabulary automaton. The word is introduced as a sequence of basic phones. The system scans the dictionary and incorporates the word model at the proper place in the state transition table. Figure 7-1 show how the word "mars" is inserted inside the state diagram given that the word "mama" is already inside the vocabulary. The word beginning and ending states are also inserted in another table so that the recognition program will identify the identity of the winning path in either the stack search or the Viterbi search.

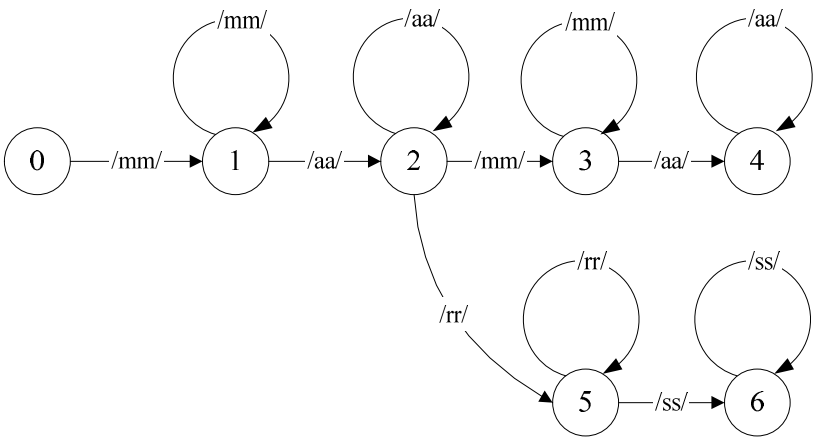


Figure 7-1 Insertion of the word "mars" inside the state diagram

7.2 Speech Segmentation

Speech segmentation consists of determining the boundaries between words, syllables, phones, etc on the continuous acoustic signal. The main difficulty of speech segmentation lies in the fact that speech is not clearly composed of distinct segments because of co-articulation [97, 99]. We have already described simple segmentation in chapter 2. Using mean magnitude and average zero crossing rate, we can label every frame as voiced, unvoiced or silence segment. Using mean magnitude ratio and average zero crossing rate, we have also developed a segmentation of speech segments into voiced non plosive, voiced plosive, unvoiced non plosive, unvoiced plosive and silence frames. However, these techniques are not directly related to the phonemic transcription of speech. The techniques of speech segmentation used in our system are based on clustering methods if the phone is not already trained. If the phone has already been trained, a Viterbi based segmentation procedure will identify the phone boundaries and the mean vector and covariance matrix will be updated.

7.2.1 Clustering Methods

We decided to use statistical clustering methods to group data obtained from a raw speech signal into similar classes. Clustering is the process of organizing data into groups [46, 49]. The resulting groups are referred to as clusters. This grouping is done according to some similarity criteria. In many cases, the criterion is some distance measure between data or group of data. Depending on the type of data, we can use different type of distance. However, the most commonly used type is the Euclidian distance. It lends itself well for our purpose because of its interpretation as a measure of proximity. There exist two families of clustering methods: the hierarchical methods and the non-hierarchical ones [46].

7.2.1.a Hierarchical Clustering

Hierarchical clustering algorithms generate a sequence of clusters by either:

- Starting with single element clusters and merging them successively into larger clusters; ultimately, we will end up with a single cluster containing all the objects. This type of algorithm is called agglomerative.

or

- Starting with the whole set of data as a single cluster and proceeding to divide it successively into smaller clusters; ultimately, we will end up

with as many clusters as objects in the set. This type of algorithm is called divisive.

The result of both approaches can be represented graphically in a form of a tree called dendrogram as shown in Figure 7-2. By cutting the dendrogram at the desired level, a clustering of the data is obtained.

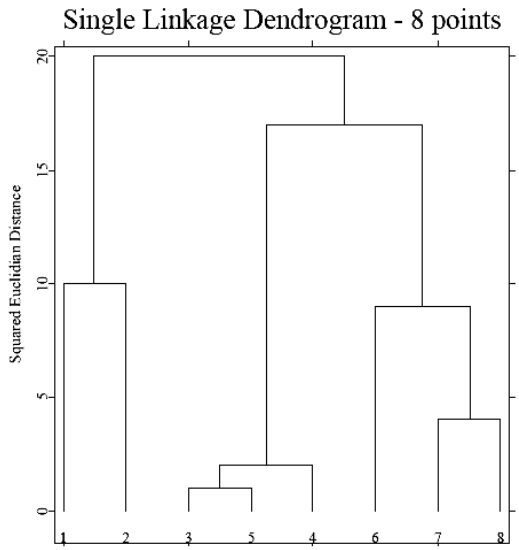


Figure 7-2 Example of a Dendrogram

We have selected the agglomerative approach in our system because we are looking at the speech signal frame after frame. Ideally, adjacent frames should cluster together. To merge clusters, we have to use some measure of distance between them. These distances are called linkages and we distinguish three different types of linkage: single, complete and average linkage as shown in Figure 7-3.

- Single linkage refers to the minimum distance $d(\mathbf{x}, \mathbf{y})$ where **x** belongs to one cluster and **y** belongs to the other one (nearest neighbor).
- Complete linkage refers to the maximum distance $d(\mathbf{x}, \mathbf{y})$ where **x** belongs to one cluster and **y** belongs to the other one (furthest neighbor).
- Average linkage refers to the average distance $d(\mathbf{x}, \mathbf{y})$ between elements of each cluster.

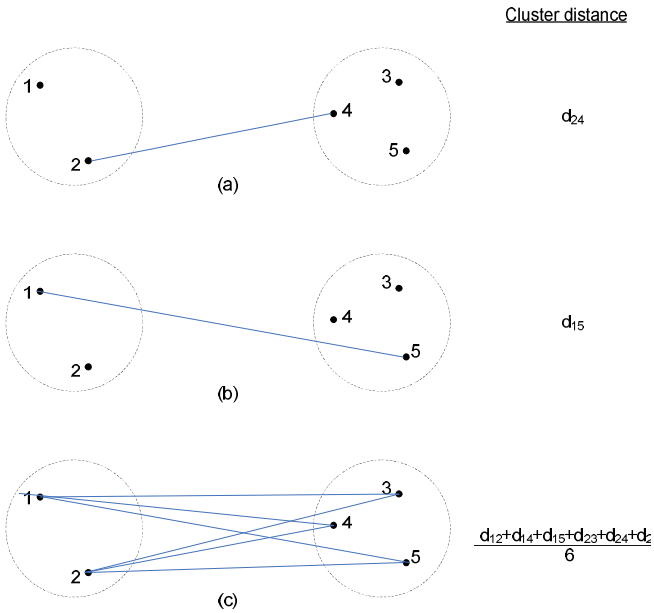


Figure 7-3 Intercluster distance for: (a) Single linkage; (b) Complete linkage; and (c) Average linkage.

7.2.1.b Non-hierarchical Clustering

Non-hierarchical methods do not proceed sequentially. The number of clusters may be specified by the user or determined during the clustering procedure. The starting step in a non-hierarchical method is either an initial partition of the objects into groups or an initial selection of seed points (cluster centroids, cluster means) forming a nucleus of clusters. One of the simplest and most widely used non-hierarchical clustering methods is the K-mean algorithm. It can be summarized as follows:

Given the data, we assume that the number of clusters is K.

- Step 1: Initialization: select K centroids (μ_1, μ_2, …, μ_K).
- Step 2: Assign each data point to the nearest centroid.
- Step 3: Recompute the new cluster centroid (by averaging).
- Step 4: Repeat step 2 and step 3 until some stopping criterion is satisfied (assignment of data does not change).

This clustering technique is quite dependent on the initial choice of centroids (step 1). If there is no a priori information, the initial centroids can be selected at random. However, we have better results if we select the centroids according to some prior knowledge. Another natural choice for the initial centroids is to select the K points that are the furthest away from each other.

7.2.2 Clustering results

Clustering is used to obtain a preliminary segmentation of the speech signal. If the segmentation is judged satisfactory, every cluster will be labelled as a phone. Assuming a Gaussian distribution for each class, initial values for the mean vector and the covariance matrix are computed. If the class w contains N elements, the initial value of the mean is:

$$\mathbf{m}_N = \frac{1}{N}\sum_{k=1}^{N}\mathbf{x}_k \quad ; \ \mathbf{x}_k \in w \tag{7.1}$$

If the number of frames N contained in the class w is much larger than the dimension p (13 in our case) of the feature vectors, then most probably the covariance matrix will not be singular. At that time, the covariance matrix can be computed the usual way.

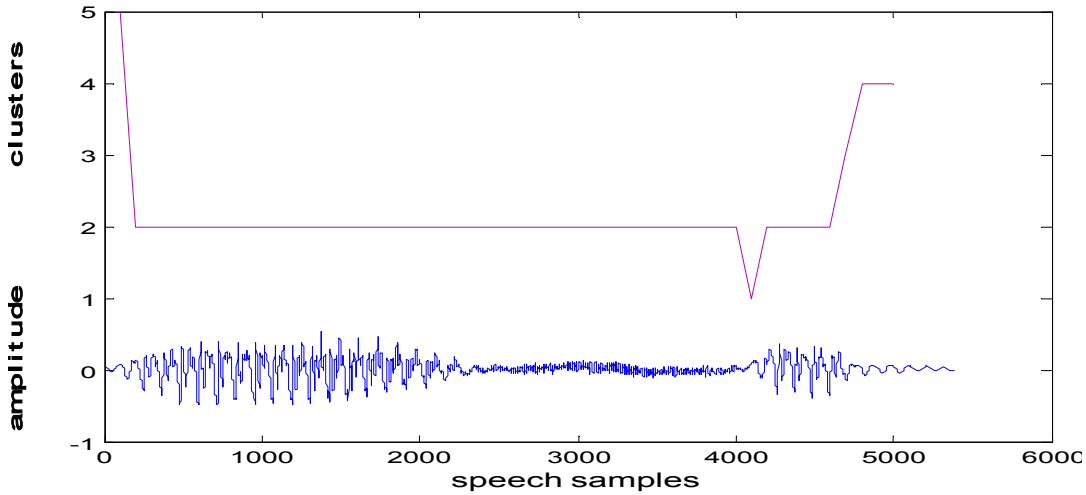
$$\mathbf{C}_N = \frac{1}{N-1}\sum_{k=1}^{N}\left(\mathbf{x}_k - \mathbf{m}_N\right)\left(\mathbf{x}_k - \mathbf{m}_N\right)^T \quad ; \ \mathbf{x}_k \in w \tag{7.2}$$

If N is smaller than p, then equation (7.2) will produce a singular matrix since it is a sum of N rank one matrices. At that time, we will make the assumption that the initial value of the covariance matrix is diagonal. The full matrix will be computed during the Viterbi updating. The diagonal of the covariance matrix is then a p dimensional vector given by:

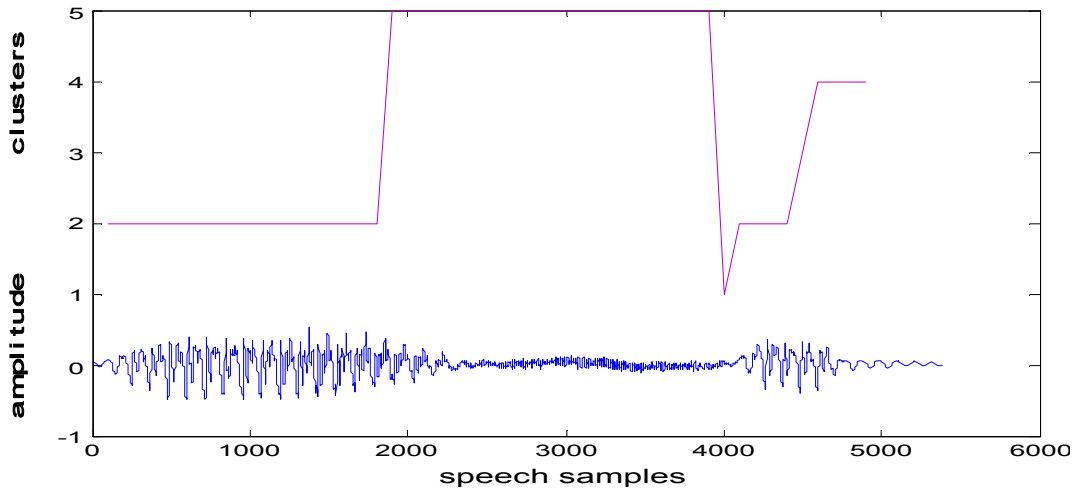
$$\left[\mathbf{C}_N\right]_{ii} = \boldsymbol{\sigma}_{iN}^2 = \frac{1}{N-1}\sum_{k=1}^{N}\left(x_{ik} - m_{iN}\right)^2 \quad ; \quad 1 \le i \le p \tag{7.3}$$

where x_{ik} is the ith component of the kth feature vector and m_{iN} is the ith component of the class mean vector.

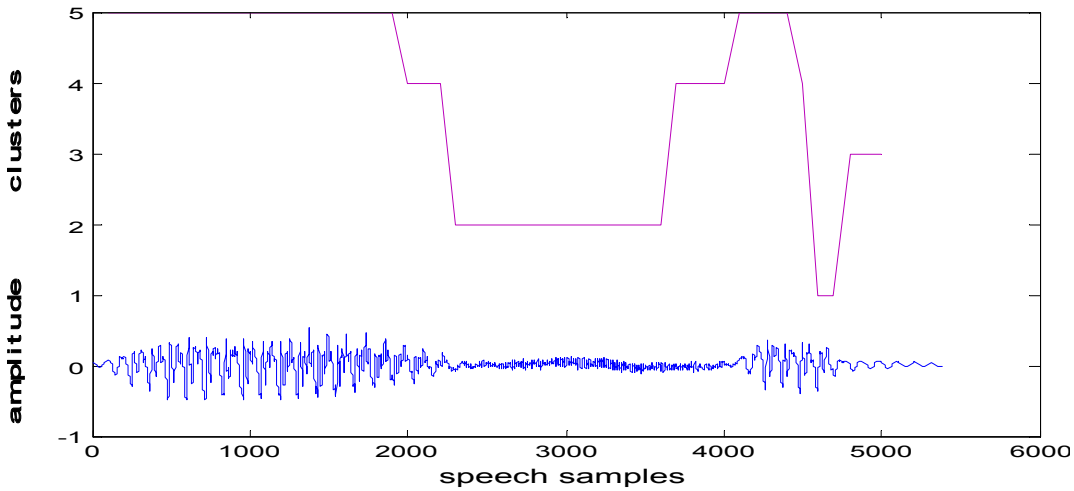
In order to be able to use the above equations, we have to select a clustering method. We are going to test first the hierarchical methods and then we will compare them with the K-mean. The agglomerative clustering methods for the word "ouahed" are presented in Figure 7-4. The output of the clustering algorithm is an integer varying between 1 and 5. The value of the integer is plotted on top of the acoustic waveform (taking into account that the frames are translated every 100 samples). The clusters are expected to be poorly separated due to co-articulation. This implies that single linkage is expected to perform weakly [46, 49]. We can remark that the average linkage provides better segmentation than either the single or the complete linkage.



(a)



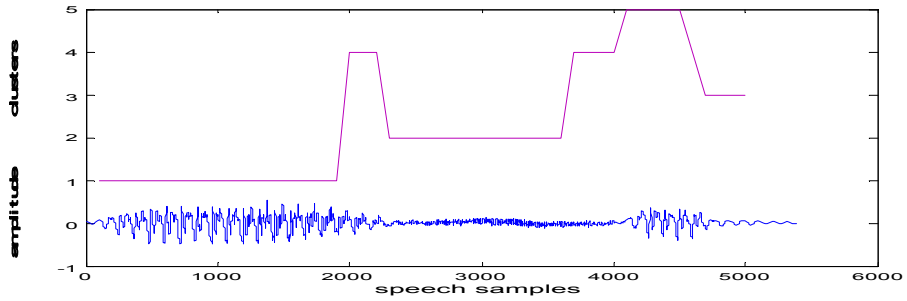
(b)



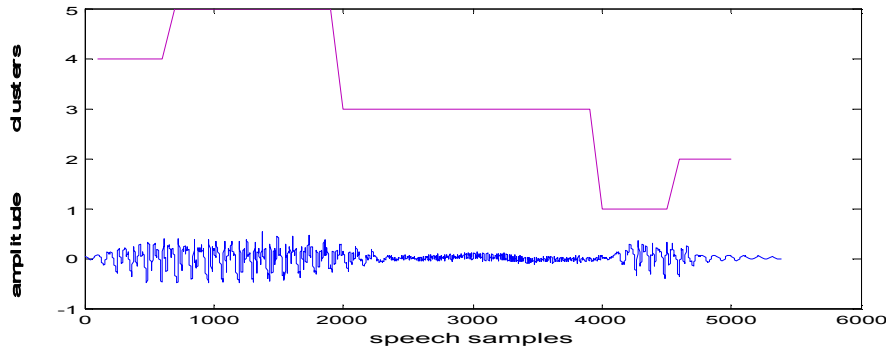
(c)

Figure 7-4 Agglomerative test: (a) Single linkage; (b) Complete linkage; (c) Average linkage

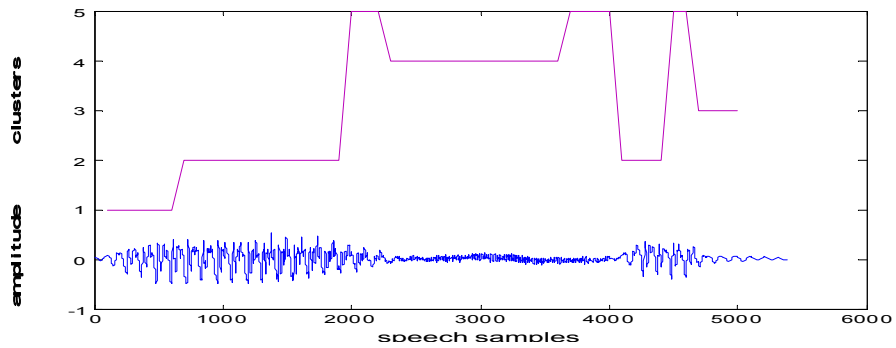
Tests conducted on other words confirm that the average linkage gives always better segmentation than the other two methods of agglomerative clustering. So, we will compare now average linkage agglomerative linkage with K-mean clustering.



(a)



(b)



(c)

Figure 7-5 Clustering of "ouahed" into 5 clusters using the K-mean with random centroids

We notice from Figure 7-5 that we obtain very different results for every run of the K-mean algorithm when the centroids are selected at random. However, a detailed observation of the three graphs shows that the results of the clustering can be exploited. However, Figure 7-6 shows clearly that the K-mean with selected centroids (inside the presumed phones) produces a segmentation that can be matched with the manual

segmentation shown in Figure 7-7. It is also observed that the K-mean produces a better segmentation that the agglomerative method with average linkage.

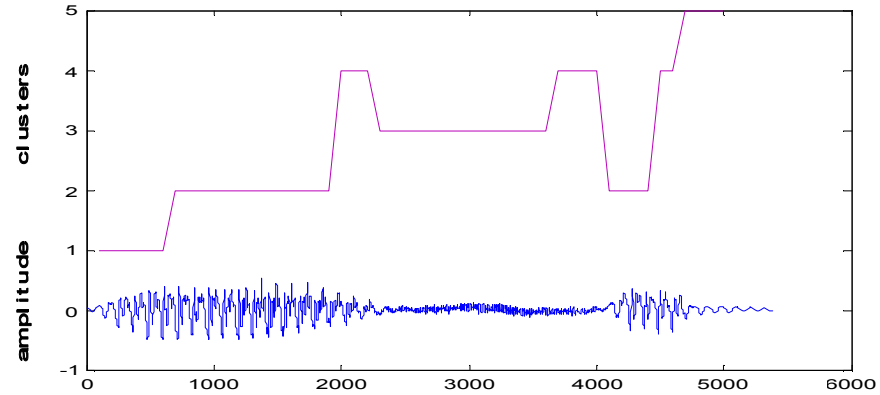


Figure 7-6 Clustering of "ouahed" into 5 clusters using the K-mean with selected centroids

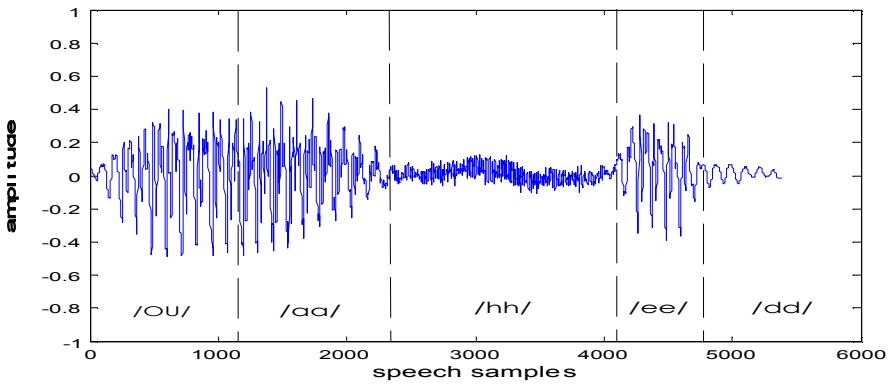


Figure 7-7 Manual Segmentation of the Word "ouahed"

Tests have been conducted with several words in the vocabulary and we have observed that the K-mean algorithm with selected centroids is the clustering method that has the best results. Therefore, if the phones composing a particular word have not been trained, initial data will be obtained using a K-mean clustering algorithm with selected centroids.

7.2.3 Segmentation using the Viterbi Algorithm

If there is initial data for every phone composing the word, the Viterbi algorithm can be used to refine and find better boundaries for the different speech segments. Once those segments have been identified, the data composing the phones will be used to update the mean vector and the covariance matrix of the phone density function. The refinement of the data is obtained as follows.

Let us call $\mathbf{m}_N$ and $\mathbf{C}_N$ the mean vector and the covariance matrix for the class w computed using N frames. The addition one more frame $\mathbf{x}_{N+1}$ will produce the following mean:

$$\mathbf{m}_{N+1} = \frac{N}{N+1}\mathbf{m}_N + \frac{1}{N+1}\mathbf{x}_{N+1} \tag{7.4}$$

We are not going to update the covariance matrix but its inverse. This is because the inverse is required in the computation of the Mahalanobis distance. In order to do so, we are going to use the Sherman-Morrison-Woodbury formula [24]. Let $\mathbf{A}$ be a non-singular matrix and $\mathbf{u}$ and $\mathbf{v}$ be two vectors, then $(\mathbf{A} + \mathbf{u}\mathbf{v}^T)$ is not singular if and only if $\sigma = 1 + \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u} \neq 0$ and:

$$\left(\mathbf{A} + \mathbf{u}\mathbf{v}^T\right)^{-1} = \mathbf{A}^{-1} - \frac{1}{\sigma}\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1} \tag{7.5}$$

Using equation (7.2), we can express the covariance matrix as:

$$(N-1)\mathbf{C}_N = \mathbf{R}_N - N\mathbf{m}_N\mathbf{m}_N^T \tag{7.6}$$

where
$$\mathbf{R}_N = \sum_{k=1}^{N}\mathbf{x}_k\mathbf{x}_k^T \tag{7.7}$$

then
$$\mathbf{R}_{N+1} = \mathbf{R}_N + \mathbf{x}_{N+1}\mathbf{x}_{N+1}^T \tag{7.8}$$

The application of equation (7.5) leads to:

$$\mathbf{R}_{N+1}^{-1} = \mathbf{R}_N^{-1} - \frac{1}{1+\mathbf{x}_{N+1}^T\mathbf{R}_N^{-1}\mathbf{x}_{N+1}}\mathbf{R}_N^{-1}\mathbf{x}_{N+1}\mathbf{x}_{N+1}^T\mathbf{R}_N^{-1} \tag{7.9}$$

Equation (7.6) has also the same shape, so (7.5) can be applied to give:

$$\mathbf{C}_{N+1}^{-1} = N\left[\mathbf{R}_{N+1}^{-1} + \frac{(N+1)}{1+(N+1)\mathbf{m}_{N+1}^T\mathbf{R}_{N+1}^{-1}\mathbf{m}_{N+1}}\mathbf{R}_{N+1}^{-1}\mathbf{m}_{N+1}\mathbf{m}_{N+1}^T\mathbf{R}_{N+1}^{-1}\right] \tag{7.10}$$

Therefore, every time a new datum is added, we can use in sequence equations (7.9) and (7.10) to update the inverse of the covariance matrix.

The Viterbi algorithm is going to be used to obtain a path in the given trellis representing the FSA associated with the word used in the training. We use the model of production described in section 6.5.1. For instance, the word "zoudj" is represented by the state diagram shown in Figure 7-8. The corresponding state trellis is presented in Figure 7-9. The optimum path followed by the Viterbi algorithm is displayed in red.

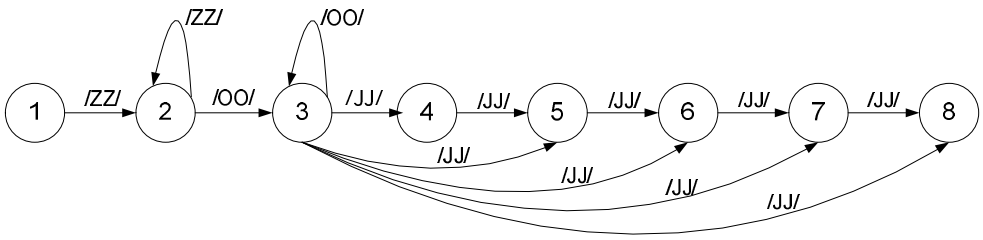


Figure 7-8 State Diagram of the Word "zoudj"

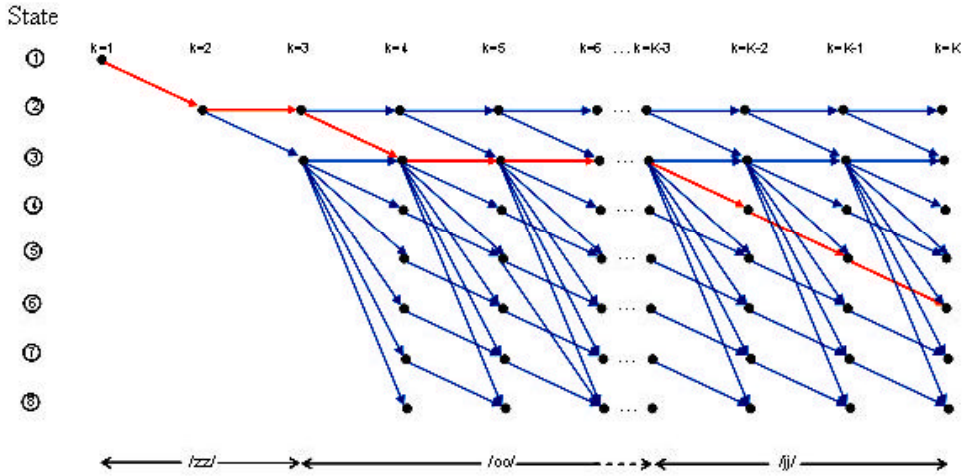
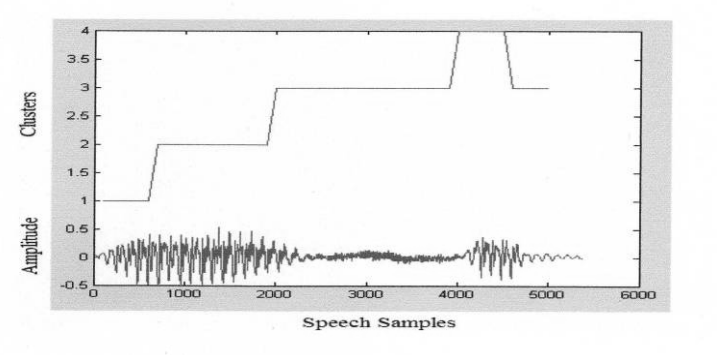


Figure 7-9 State Trellis of the Word "zoudj"

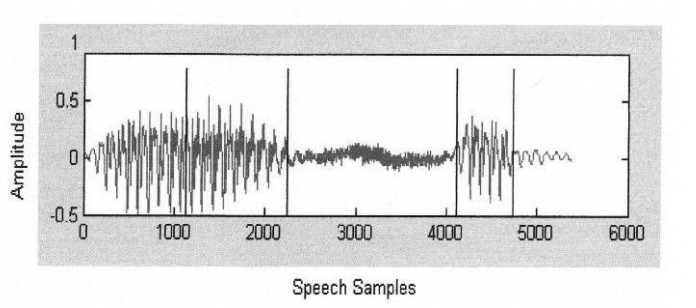
The form of the Viterbi algorithm used in this section is slightly different from the one used to decode HMM's. We use the path metric (6.15). The algorithm used is directly the one given by Forney in reference [33]. The state q_k at frame time k is a finite number m from N different states; $1 \leq m \leq N$. Among all the paths passing by the node q_k, at time k, the one with the largest path metric is called the survivor terminating at q_k and is denoted $S(q_k)$. At any time $k > 0$, there are N survivors, one for each q_k. The optimum path must begin with one of these survivors. At any time k, we must remember the N survivors and their metric. Along with the metric, we must also keep track of the path labels.

Once the optimum path found, a backtracking algorithm is used to obtain the label of every frame of the speech segment. For example, the word "zoudj" will produce the sequence: /zz/ /zz/ …/zz/ /oo/ /oo/ …/jj/ /jj/ /jj/ /jj/. Using this labelling, we can identify the phone boundaries and identify the different segments of the speech signal. We can then use equations (7.4), (7.9) and (7.10) to update the preliminary data

obtained by the K-mean clustering. In order to minimize the probability of using an outlier during the update process, we use the fact that the square Mahalanobis distance d^2 is a chi-square random variable with p degrees of freedom (p = 13 in our case) when we are on the correct path. The chi-square distribution with 13 degrees of freedom will give the probability $P[d^2 > 29.83] = 0.005$ (0.5%). So, on a correct path, the value of the local path metric is $L(\mathbf{x}) = p - d^2$ and it is larger than -16.82 with a probability of 99.5%. Therefore, in a given segment labelled w, a frame with $L(\mathbf{x}) > -16.82$ can be safely used to update the mean vector and the covariance matrix.



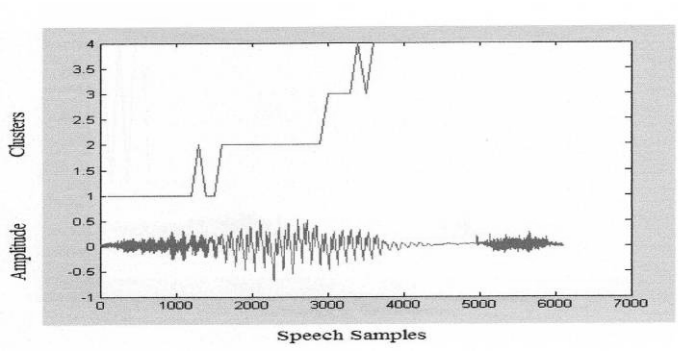
(a)



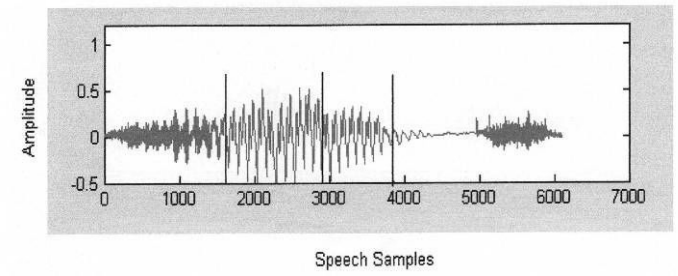
(b)

Figure 7-10 Segmentation of the word "ouahed" using (a) the K-mean, (b) refined segmentation using Viterbi Algorithm

Figure 7-10 shows the result of Viterbi segmentation on the word "ouahed". We see clearly the different phones that compose the word: /oo/ /aa/ /Ha/ /Dd/. The corresponding speech data are used to update the parameters corresponding to each phoneme. This automatic segmentation has detected the voiced-voiced transition around sample 1100 between the phone /oo/ and the phone /aa/. This transition is very difficult to detect manually.

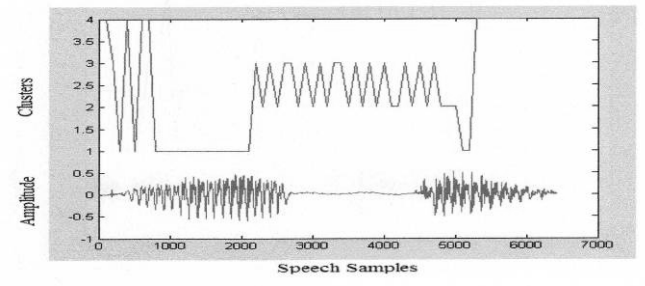


(a)

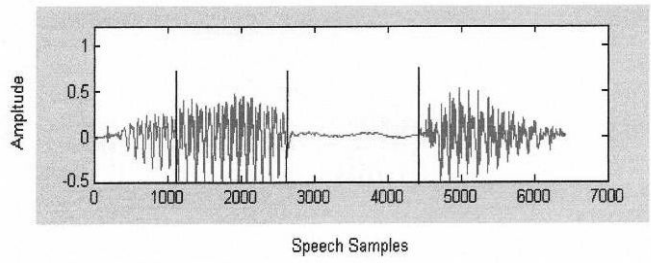


(b)

Figure 7-11 Segmentation of the word "zoudj" using (a) the K-mean, (b) refined segmentation using Viterbi Algorithm

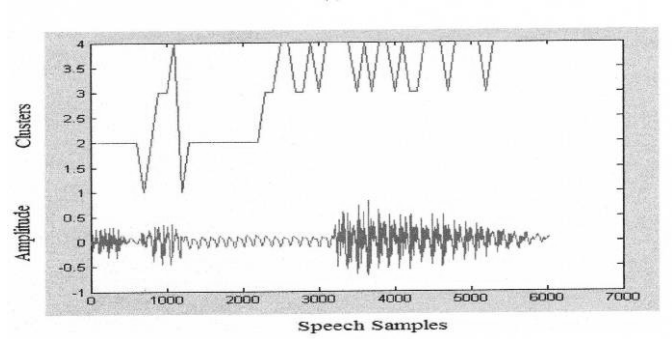


(a)

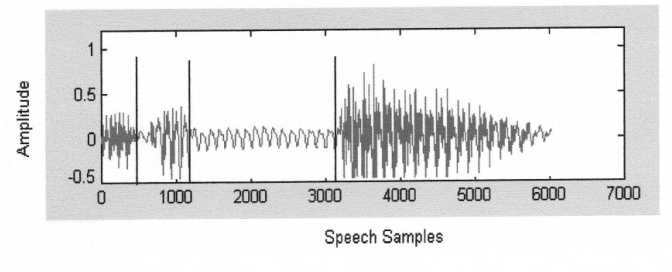


(b)

Figure 7-12 Segmentation of the word "tleta" using (a) the K-mean, (b) refined segmentation using Viterbi Algorithm



(a)



(b)

Figure 7-13 Segmentation of the word "reb3a" using (a) the K-mean, (b) refined segmentation using Viterbi Algorithm

Figure 7-11, Figure 7-12 and Figure 7-13 show the result of Viterbi segmentation for three other words ("zoudj", "tleta", "reb3a"). Here also, the resulting segmentation is better than a segmentation that would be performed by hand. We remark also that the Viterbi refinement corrects the segmentation errors that are introduced by the K-mean algorithm. The only problem with our segmentation procedure is the fixed number of classes (phones) that must be user defined. However, the training becomes completely automatic and the user is freed from the tedious task of labelling every speech segment. The stack recognition algorithm developed in chapter 6 did not use our training method for its initial vocabulary. Yet, we are using the training method for adding new words in the automaton defining the vocabulary.

Chapter 8

Conclusion

Speech recognition has entered our every day life. Dictation software programs, modern telephone sets, even cars are examples of this trend. The new systems are more and more sophisticated. Most of them are oriented toward continuous speech recognition. In previous years, the research in speech used to employ complex and costly systems whereas actual systems are more oriented toward immediate use. Our work in this thesis has essentially the same objective. We wanted to develop a continuous speech recognition system that uses standard hardware such as personal computer sound card as acquisition system.

The stack algorithm developed in this thesis has provided very satisfactory results. The obtained recognition rate is on the same level as the one obtained using hidden Markov models. We obtained this result despite the fact that we did not make use of derivative or acceleration parameters. The algorithm presents the advantage of depending only linearly on the number of states representing the vocabulary. This is quite advantageous in large size vocabularies and complex scripts. Another advantage of the algorithm is the fact that it does not need any resetting when a word is recognized. This is essentially due to the adopted path metric. Another particularity of the recognition system is the assignment of multiple unimodal (Gaussian) distributions with the same label instead of a single multimodal one.

While collecting data for training both our algorithms, we were faced with typical defects of PC sound cards and low cost microphones. We have then developed a novel method for clipped speech restoration. This method has proved to be very efficient. The recovered speech can be used safely with either the training or the recognition part of our system. It can also be used as a stand alone program for enhancing the intelligibility of acquired speech.

The defect of using a single microphone is that the microphone picks useful speech signal and unwanted noise. We have developed time invariant and time varying Wiener filters for denoising the acquired speech. The time varying filter has produced remarkable results. These results have been obtained with very low order filters (two).

We have shown in chapter four that this is due to a time varying gain that depends on the region of the speech segment we are in. Our denoising algorithm can be used in real time application, such as telephony. This is due to fact that the algorithm requires just a small delay (of the size of the noise-only section) to become operational.

The recognition algorithms used in our thesis are all based on the assumption that the parameters representing speech are Gaussian. Among three different sets, we have demonstrated that the MFCC parameters are the only ones that have this Gaussian behavior. We have also shown that they provide the best separation between classes.

Finally, the segmentation and training method developed for our stack algorithm frees the user from the tiresome job of manually segment and label speech segments for training.

Suggestion for Further Research

The hidden Markov model methods provide very powerful design means for continuous speech recognition. We suggest imbedding HMM phoneme models inside the total finite state vocabulary. At that time, the stack algorithm will work in a top down manner, starting from the sentence syntax all the way to the phoneme syntax. It is evident that the path metric should change. This is the object of this further research.

We have analysed three different sets of parametric representation of speech: LPC, PARCOR and MFCC. It is suggested to compare other sets. Another possibility is the use of nonlinear transformations for the LPC parameters and the PARCOR coefficients. For example, the PARCOR coefficients are numbers that have a magnitude bounded by one, $|k_i| < 1$. They have statistical properties that are similar to correlation coefficients. We should test Fisher's transformation [28, 29] on PARCOR coefficients. This transformation is commonly used to map correlation coefficients to near Gaussian random variable. The LPC parameters on the other hand can be transformed to near Gaussian variables using Box-Cox transformations [13] and the resulting variables tested statistically.

In the training part of our system, we have used the K-mean clustering algorithm. This method requires the knowledge of the number of clusters in advance. It is suggested to use a method like the ISODATA [4]. This method does not require any previous information on the number of clusters.

References

[1] Akilal, K. and H. Aroune, *Pattern Recognition Techniques Applied to Speech Recognition,* Final Year Project, DGEE, FSI, Univ. de Boumerdes, 2001.

[2] Albright, R.W., *The International Phonetic Alphabet: Its Backgrounds and Development* (Publication Seven of the Indiana University Research Center in Anthropology, Folklore, and Linguistics, 1958); also Part III of International Journal of American Linguistics, Vol 24, No 1. B. 1958

[3] Atal, B.S. and S.L. Hanauer, "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave", *Jour. of the Acoustical Society of America*, Vol.50, pp.637-655, 1972.

[4] Ball, G.H. and D.J. Hall, "Some Fundamental Concepts and Synthesis Procedures for Pattern Recognition Preprocessors", *Proc. of the International Conf. on Microwaves, Circuit Theory and Information Theory*, Tokyo, Japan, Sept.1964.

[5] Baum, L.E., "An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of a Markov Process", *Inequalities*, Vol.3, pp.1-8, 1972.

[6] Baum, L.E. and J.A.Eagon, "An Inequality with Applications to Statistical Estimation for probabilistic Functions of Markov Processes and to a Model for Ecology", *Bulletin of American Mathematical Society*, Vol.73, pp.360-363, 1967.

[7] Baum, L.E. and T. Petrie, "Statistical Inference for Probabilistic Functions of Finite State Markov Chains", *Annals of Mathematical Statistics*, Vol.37, pp.1554-1563, 1966.

[8] Baum, L.E., T. Petrie, G. Soules and N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains", *Annals of Mathematical Statistics*, Vol.41, n°1, pp.164-171, 1970.

[9] Baumann, R.H., J.C.R. Licklidler and B. Howland, "Electronic Word Recognizer", *Jour. of the Acoustical Society of America*, Vol.26, p.137, 1954.

[10] Bellman, R., *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.

[11] Blum, J. R. and J. I. Rosenblatt, *Probability and Statistics*, W. B. Saunders Company, 1972.

[12] Bogert, B.P., M.J.R. Healy, and J.W. Tukey, "The quefrency alanysis of times series for echos: Cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking", in *Proceedings of the Symposium on Time Series Analysis*, (M. Rosenblatt, ed.), New York: John Wiley and Sons, Inc., 1963.

[13] Box, G. E. P. and D. R. Cox, "An Analysis of Transformations", *Journal of the Royal Society* (B), 26, n°2, pp.211-255, 1964.

[14] Brakta, N. and M. Hadibi, *Pattern Recognition Techniques Applied to Speech Recognition,* Final Year Project, INELEC, 1999.

[15] Brakta, N., A. Dahimene and M. Noureddine, "Pattern Recognition Techniques Applied to Speech Recognition", *CGME'*01, Constantine, 2001.

[16] Cambridge University Engineering Department, *The HTK Book*, Cambridge University, 2002.

[17] Chandra, S. and W. C. Lin, "Experimental Comparison between Stationary and Non Stationary Formulations of Linear Prediction Applied to Voiced Speech Analysis", *IEEE trans. of ASSP*, Vol.22, n°6, Dec. 1974.

[18] Crochiere R.E. and L.R. Rabiner, "Interpolation and decimation of digital signals – a tutorial review", *Proceeding of the IEEE*, Vol.69, pp.300–331, 1981.

[19] Dahimene, A., M. Noureddine and A. Azrar, "A simple Algorithm for the Restoration of Clipped Speech Signal", *Informatica*, Vol. 32, n°2, pp. 183-188, 2008.

[20] Dahimene, A., M. Noureddine and A. Azrar, "Sequential Decoding Applied to Speech Recognition", to be published, *Informatica.*

[21] David, E. and O. Selfridge, "Eyes and Ears for Computers", *Proc. of the IRE*, pp.1093-1101, May 1962.

[22] Davis, H., R. Biddulph and S. Balashek, "Automatic Recognition of Spoken Digits", *Jour. of the Acoustical Society of America*, Vol.24, n°6, pp.637-642, Nov. 1952.

[23] Davis, S.B. and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE*

Transactions on Acoustics, Speech and Signal Processing, vol. 28, pp. 357–366, August 1980.

[24] Dennis, J.E. and R.E. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, 1983.

[25] Dudley, H. and T.H. Tarnoczy, "The Speaking Machine of Wilhelm Von Kempelen", *Jour. of the Acoustical Society of America*, Vol. 22, 1950.

[26] IBM-Embedded ViaVoice, Software,

http://www- 306.ibm.com/software/pervasive/embedded_viavoice/, 2008

[27] Fano, R. M., "A Heuristic Discussion of Probabilistic Decoding", *IEEE Trans. Inform. Theor.,* vol. IT-9, pp. 64-74, 1963.

[28] Fisher, R. A., "The Statistical Utilization of Multiple Measurements", *Annals of Eugenics*, 8, pp. 376-386, 1938.

[29] Fisher, R. A., "The Use of Multiple Measurements in Taxonomic Problems", *Annals of Eugenics*, 7, pp. 179-188, 1936.

[30] Fisher, W.M., G.R. Doddington and K.M. Goodie-Marshall, "The DARPA Speech Recognition Database: Specifications and Status", *Proc. of DARPA Workshop on Speech Recognition*, pp.93-99, 1986.

[31] Flanagan, J. L., *Speech Analysis, Synthesis and Perception*, Springer-Verlag, 1972.

[32] Flanagan, J. L., C. H. Coker, L. R. Rabiner, R. W. Schafer, and N. Umeda, "Synthetic voices for computers", *IEEE Spectrum*, vol. 7, pp. 22–45, October 1970.

[33] Forney, G.D., "The Viterbi Algorithm", *Proceeding of the IEEE*, Vol.61, n°3, pp268-279, March 1973.

[34] Fosler-Lussier, E., *Markov Models and Hidden Markov Models: A Brief Tutorial*, ICSI, tech. rep. TR-98-041, 1998.

[35] Fukunaga, K., *Introduction to Statistical Pattern Recognition*, 2nd ed., Academic Press, 1990.

[36] *Fundamentals of the Human Speech Production*, http://www.student.chula.ac, 2006

[37] Gabor, D., "Theory of Communication", *Journ. of IEE*, Vol.93, n°III, Nov. 1946.

[38] Goldstine, H. H. and A. Goldstine, *The Electronic Numerical Integrator and Computer (ENIAC)*, 1946 (reprinted in *The Origins of Digital Computers: Selected Papers*, Springer-Verlag, New York, 1982, pp. 359-373)

[39] Gueguen, C., J., and Carayannis, G., "Analyse de la Parole par Filtrage Optimal de Kalman", *Automatisme*, Tome XVIII, pp.99-105, March 1973.

[40] *The Handbook of the International Phonetic Association*, Cambridge University Press, pp 194-197, 1999.

[41] Harakat, M., *Acoustic and Phonology*, Dar-El-Affak, 1999

مصطفى حركات, الصوتيات و الفونولوجيا-دار الآفاق, 1999

[42] Heute, U., "Noise Reduction", in Hänsler, E. and G. Schmidt, Ed., *Topics in Acoustic Echo and Noise Control*, Springer, Berlin, New York, 2006.

[43] Hoffman, J.D., *Numerical Methods for Engineers and Scientists*, 2nd ed., Marcel Dekker, 2001.

[44] Itakura, F. and S. Saito, "Analysis synthesis telephony based upon the maximum likelihood method", *Reports of* 6th *Int. Cong. Acoust.*, ed. by Y. Kohasi, Tokyo, C-5-5, C17–20, 1968.

[45] Itakura, F. and S. Saito, "Analysis synthesis telephony based on the partial autocorrelation coefficient", *Acoust. Soc. of Japan Meeting*, 1969.

[46] Jain, A.K. and R.C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[47] Jakobson, R., G. Fant and M. Halle, *Preliminaries to Speech Analysis: The Distinctive Features and their Correlates*, MIT Press, Cambridge, Mass. 1952.

[48] Jelinek, F., "A Fast Sequential Decoding Algorithm Using Stack", *IBM J. Res. Dev.,* vol.13, pp. 675-685, 1969.

[49] Johnson, R. A. and D. W. Wichern, *Applied Multivariate Statistical Analysis*, 3rd Ed., Prentice Hall, 1982.

[50] Kaddouri, S., *Speech Denoising using Time-Varying Wiener Filter*, Final Year Project, DGEE, FSI, Univ. de Boumerdes, 2008.

[51] Kaddouri, S. and A. Dahimene, "Speech Denoising using Time-Varying Wiener Filter, *Informatica*, to be published.

[52] Koenig, R., H.K. Dunn and L.Y. Lacy, "The Sound Spectrograph", *Jour. of the Acoustical Society of America*, Vol.18, pp.19-49, 1946.

[53] Kohavi, Z., *Switching and Finite Automata Theory*, Mc Graw Hill, 1970.

[54] Lawson, C.L., and R.J. Hanson, *Solving Least Squares Problems*, Prentice Hall, 1974.

[55] Makhoul, J., "Linear Prediction, A Tutorial Review", *Proceeding of the IEEE*, Vol.63, n°4, pp. 561-580, April 1975.

[56] Markel, J.D., "The Prony Method and its Applications to Speech Analysis", *Jour. of the Acoustical Society of America*, Vol. 55, pt. 1, p.105 (A), Jan. 1971.

[57] Markel, J.D., "Digital Inverse Filtering – a New Tool for Formant Trajectory Estimation", *IEEE Trans. Audio Electroacoust.*, Vol.AU-20, pp.129-137, June 1972.

[58] Markel, J.D. and Jr.A.H. Gray, *Linear Prediction of Speech*, Springer-Verlag, Berlin, 1976.

[59] Moharir, G., *Spectral Subtraction Method for Speech Enhancement*, M. Tech. Thesis, Department of Electronic Engineering, I.I.T., Bombay, India, Jan. 2002.

[60] Mouhamou, R. and D. Iadaden, *Pattern Recognition Techniques Applied to Speech Recognition,* Final Year Project, DGEE, FSI, Univ. de Boumerdes, 2002.

[61] Mouhouche, B. and S. Gunadiz, *Pattern Recognition Techniques Applied to Speech Recognition,* Final Year Project, DGEE, FSI, Univ. de Boumerdes, 2001.

[62] Nilsson, M. and M. Ejnarsson, *Speech Recognition using Hidden Markov Models, Performance Evaluation in NoisyEnvironment*", Master degree thesis, Blekinge Institute of Technology, Sweden, March 2002.

[63] Nuance, Dragon Naturally Speaking, http://www.nuance.com/naturallyspeaking/, 2008.

[64]Oppenheim, A.V., *Superposition in a class of nonlinear systems*, PhD dissertation, MIT, 1964. Also: MIT Research Lab. of Electronics, Cambridge, Massachusetts, Technical Report No. 432, 1965.

[65] Oppenheim, A.V, R.W. Schafer and T.G. Stockham Jr., "Nonlinear filtering of multiplied and convolved signals," *Proceedings of IEEE*, vol. 56, no. 8, pp. 1264–1291, August 1968.

[66] Oppenheim, A.V. and R.W. Schafer, *Digital Signal Processing*, Prentice-Hall, 1975.

[67] O'Shaughnessy, D., *Speech Communications: Human and Machine*, Addison-Wesley, 1987.

[68] Papoulis, A., *Probability, Random Variables and Stochastic Processes*, 3rd ed., Mc Graw Hill, 1991.

[69] Philips, G.M., *Interpolation and Approximation by Polynomials*, Springer-Verlag, New-York, 2003.

[70] Rabiner, L.R. and R.W. Schaffer, *Digital Processing of Speech Signals,* Prentice Hall, 1978.

[71] Rabiner, L.R. and S.E. Levinson, "Isolated and Connected Word Recognition: Theory and Selected Applications", *IEEE trans. Communications*, Vol.29, n°5, pp.621-659, May 1981.

[72] Rabiner, L.R. and B.H. Juang, "An Introduction to Hidden Markov Models", *IEEE ASSP Magazine*, pp.4-16, Jan. 1986.

[73] Rabiner, L.R., "Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceeding of the IEEE*, Vol.77, n°2, pp. 257-286, 1989.

[74] Rabiner, L.R., *Fundamentals of Speech Recognition*, Prentice Hall, 1993.

[75] Ravichandran, T., "Performance Enhancement on Voice using VAD Algorithm and Cepstral Analysis", *Journal of Computer Science*, pp.835-840, 2006.

[76] Reggab, M, *Continuous Speech Recognition using Hidden Markov Models: Application to colloquial Algerian Arabic*, Magister thesis, DGEE, FSI, Univ. de Boumerdes, 2004.

[77] Rosenberg, A.E., "Effect of the Glottal Pulse Shape on the Quality of Natural Vowels", *Jour. of the Acoustical Society of America*, Vol.49, n°2, pp.583-590, Feb.1971.

[78] Rosenberg, A.E. and F. Itakura, "Evaluation of an automatic word recognition system over dialed-up telephone lines", *Jour. of the Acoustical Society of America*, suppl.I, Vol. 60, p. S12, 1976.

[79] Sakoe, H. and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition", *IEEE Trans. Acoust., Speech and Signal Proc.*, Vol.26, pp.43-49, Feb. 1978.

[80] Scagliola, C. and L. Marni, "A Continuous Speech Recognition System Based on a Diphone Spotting Approach", in *Cybernetic Systems: Recognition, Learning, Self-Organisation,* edited by Caianiello, E. R. and Musso, G., Research Study Press, John Wiley, 1984.

[81] Scowen, R.S., "Extended BNF – A Generic Base Standard", *Software Engineering Standard Symposium*, 1993.

[82] Shannon, C.E., "Communication in the Presence of Noise", *Proceeding of the IRE*, Vol.37, n°1, pp.10-21, jan. 1949.

[83] Smith, C.P., "A phoneme detector", *Jour. of the Acoustical Society of America*, Vol.23, pp.446-451, 1951.

[84] Srinath, M.,D. and P., K. Rajasekaran, *An Introduction to Statistical Signal Processing with Applications*, John Wiley & Sons, 1979.

[85] Stevens, S.S. and J. Volkman, "The relation of pitch to frequency", *American Journal of Psychology*, vol. 53, p. 329, 1940.

[86] Stewart, G.W., *Introduction to Matrix Computations*, Academic Press, 1973.

[87] Swadesh, M., "The Phonemic Principle", *Language*, Vol.10, pp.117-129, 1934.

[88] Tou, J. T. and R. C. Gonzalez, *Pattern Recognition Principles*, Addison Wesley Publishing Co., 1974.

[89] Twaddell, W.F., "On Defining the Phoneme", Language monograph n°16, *Language*, 1935.

[90]Ungerboeck, G., "Trellis-coded modulation with redundant signal sets part I: introduction," *IEEE Communications Magazine*, vol. 25-2, pp. 5-11, 1987.

[91]Van Trees, H. L., *Optimum Array Processing*, Wiley, New York, 2002.

[92] Vaseghi, S. V., *Advanced Signal Processing and Digital Noise Reduction*, John Wiley and Teubner, 1996.

[93] Vintsyuk, T.K., "Speech Discrimination by Dynamic Programming", *Kibernetka (Cybernetics)*, Vol.4, pp.81-88, 1968.

[94] Vintsyuk, T.K., "Element-wise Recognition of Continuous Speech Consisting of Words from a Specific Vocabulary", *Kibernetica (Cybernetics)*, pp.133-143, 1971.

[95] Viterbi, A.J., "Error Bounds for Convolutional Codes and Asymptotically Optimum Decoding Algorithm", *IEEE Trans. on Information Theory*, IT-13, April 1967.

[96] Viterbi, A. J. and J. K. Omura, *Principles of digital Communication and Coding*, Mc Graw Hill, 1979.

[97] Vorstermans, A., J.P. Martens and B. Van Coile, "Automatic Segmentation and Labeling of Multi-Lingual Speech Data", *Speech Comm.*, n°19, pp.271-293, 1996.

[98] Welch, L.R., "Hidden Markov Models and the Baum-Welch Algorithm", *IEEE Information Theory Society Newsletter*, Vol.53, n°4, Dec.2003.

[99] Wang, D., L. Liu and H.J. Zhang, "Speech Segmentation without Speech Recognition", *ICASSP*, 2003.

[100] Wiener, N., *Extrapolation, Interpolation and Smoothing of Stationary Time Series*, MIT Press, Cambridge, Mass., 1949.

[101] Wilks, S.S., *Mathematical Statistics*, John Wiley, 1963.

[102] Wozencraft, J. M., "Sequential Decoding for Reliable Communications", *IRE Nat. Conv. Rec.,* vol.5, pt.2, pp.11-25, 1957.

[103] Yang, W.Y., W. Cao, T.S. Chung and J. Morris, *Applied Numerical Methods using MATLAB*, John Wiley, 2005.

[104] Zebo, T.J. and W.C. Lin, "On the Accuracy of Formant Parameter Estimation Based on the Method of Prony", *Int. Conf. Speech Communication and Processing*, Paper B10, pp.85-88, 1972.

[105] Zigangirov, K. S., "Some Sequential Decoding Procedures", *Problemy Peredachi Informatsii,* vol.2, pp. 13-25, 1966.

[106] Zwicker, E. and H. Fastl, *Psycho-acoustics*, Springer-Verlag, 2nd Edition, 1990.

AppendixA

The Adopted List of Phonemes

Number	Arabic	Symbol	Pronounciation
1	ا	aa	*The vowel /a/ following consonant or* at the beginning of a word.
2	ب	bb	As in English
3	ت	tt	A soft *dental,* like the Italian */t/*
4	ث	st	Very nearly the sound of */th/* in *thing.*
5	ج	Dj	As in English
6	ح	Ha	A strong Aspirate
7	خ	Kh	*Guttural,* like the Scotch */ch/* in *loch.*
8	د	Dd	Soft *dental*
9	ذ	th	A sound between */dh/* and */z/,* very near to */th/* in *this*
10	ر	rr	As in English
11	ز	zz	As in English
12	س	ss	As in English
13	ش	sh	As in English
14	ص	sa	A strongly articulated */s/.*
15	ض	da	A strongly articulated /d/
16	ط	ta	A strongly articulated palatal */t/.*
17	ظ	dh	A strongly articulated /th/
18	ع	3a	A *guttural,* that is close to */aa/*.
19	غ	gh	The sound *r* in french.
20	ف	ff	As in English
21	ق	qa	
22	ك	kk	As in English
23	ل	ll	As in English
24	م	mm	As in English
25	ن	nn	As in English
26	هـ	hh	As in English
27	و	wa	As in English
28	ي	ya	As in English
29	ئ، ؤ، أ ، ء	ia	/aa/ ,/ii/,/oo/ preceeded by a vowel.
30	-	ii	As in Italian
31	ِ	oo	As in Italian
32		aa ita	As in Italian

Table A-1 The Adopted List of Phonemes [41]