NOTE TO USERS

uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Lech Szymanski
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.A.Sc. (Electrical Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Comb Filter Decomposition Feature Extraction for Robust Automatic Speech Recognition

TITRE DE LA THÈSE / TITLE OF THESIS

M. Bouchard
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

R. Goubran

W. Gueaieb

Gary W. Slater
LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES / DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

COMB FILTER DECOMPOSITION FEATURE EXTRACTION FOR ROBUST AUTOMATIC SPEECH RECOGNITION

by

Lech Szymanski

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Applied Science

Ottawa-Carleton Institute For Electrical and Computer Engineering

School of Information Technology and Engineering

University of Ottawa

Ottawa Ontario

2005

Your file Votre référence
ISBN: 0-494-11423-1
Our file Notre référence
ISBN: 0-494-11423-1

ABSTRACT

This thesis discusses the issues of Automatic Speech Recognition in presence of additive white noise. Comb Filter Decomposition (CFD), a new method for approximating the magnitude of the speech spectrum in terms of its harmonics is proposed. Three feature extraction methods from CFD coefficients are introduced. The performance of the method and resulting features are evaluated using simulated recognition systems with Hidden Markov Model classifiers and conditions of additive white noise under varying Signal to Noise ratios. The results are compared with the performance of the existing robust feature extraction methods. The results show that the proposed method has a good potential for Automatic Speech Recognition under noisy conditions.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Dr. Martin Bouchard, for his direction and support, without which this work would not be possible and to SoftdB (http://www.softdb.com) for supporting this research. Special thanks to Dave Fenton for inspiring me to take interest in DSP.

TABLE OF CONTENTS

LIST OF FIGURES

GLOSSARY

ACFD – Autocorrelation normalization/Comb Filter Decomposition

ANN – Artificial Neural Network

ASR – Automatic Speech Recognition

BP – Backpropagation

CC – Cepstrum Coefficients

CFD – Comb Filter Decomposition

DB – Decibels

DBN – Dynamic Bayesian Network

DFT – Discrete Fourier Transform

DSP – Digital Signal Processing

DCT – Discrete Cosine Transform

EM – Expectation Maximization

FFT – Fast Fourier Transform

FIR – Finite Impulse Response

FR – Frequency Response

GMM – Gaussian Mixed Model

HMM – Hidden Markov Model

HOS – Higher Order Statistics

IR – Impulse Response

LSF – Line Spectral Frequency

MPR – Magnitude-Phase Reinforcement

MFCC – Mel-frequency Cepstrum Coefficients

ML – Maximum Likelihood

MLP – Multilayer Perceptron Network

NN – Neural Network

LPC – Linear Prediction Coding

PLP – Perceptive Linear Prediction

RASTA – Relative Spectral Processing

RNN – Recursive Neural Network

RTRL – Real-Time Recurrent Learning

SNR – Signal to Noise Ration

STFT – Short-Time Fourier Transform

SVM – Support Vector Machine

NOTATION

All the math shown in this thesis is in discrete time, where n is the sample number. For sampling theory basics please refer to [4]. The bold symbols signify vectors and matrices, for which the dimensions, if not stated, should be apparent from the context. This work attempts to use symbols consistently throughout all the sections, but some might change their meaning for different contexts. Below is the symbol list with the associated meaning for a given context:

$*$ - convolution operator
(as superscript) conjugation operator
$|\ |$ - magnitude operator
$\mathbf{A}$ - state transition matrix
$A(z)$ - denominator of LPC vocal tract model
$a_t(k)$ - k^{th} LPC coefficient for frame t
a_{ij} - probability of transition from state i to state j
$\arg\{\ \}$ - argument operator
$\mathbf{b}_{\mathbf{x}_t}$ - observation probability vector for frame t
$b_i(\mathbf{x}_t)$ - i^{th} coefficient of observation probability vector for frame t
$\mathbf{B}$ - observation probability distribution matrix
$c_t(k)$ - k^{th} cepstral coefficient for frame t
c_{kg} - mixing coefficient for the g^{th} Gaussian of the k^{th} state
dB - (Signal spectral context) $dB = 20\log_{10}(a)$, where 'a' is a magnitude spectrum value at a given frequency
(SNR context) $dB = 10\log_{10}(a)$, where 'a' is the ratio of signal to noise variance
$\det(\ )$ - determinant operator
$E[\ ]$ - expectation operator
$E_t^k(k)$ - k^{th} spectral entropy coefficient of the K-region analysis for frame t
f - frequency
$f_v(\ )$ - nonlinear vocal tract configuration function
g - index variable
G - (GMM context) number of Gaussian distribution in GMM model
G_n - Gain at sample n
h(n) - convolutional noise
i - index variable
j - (FFT context) square root of -1
- (as subscript) index variable
k - index variable

K - (CFD context) number of delays in CFD analysis
(HMM context) number of pronunciation of a word of given class
(Spectral entropy context) number of regions in spectrum

L - (Cepstral feature adjustment context) number of frames used for normalization
(HMM context) number of frames in a word pronunciation
(Short-time spectral analysis context) frame overlap in samples

L^k - number of frames in k^{th} pronunciation of a word

λ - HMM model

m - frequency sample

M - total number of phoneme labels in ASR alphabet

$\max[\ ]$ - maximum operator

$\min[\ ]$ - minimum operator

μ_{kg} - mean vector for g^{th} Gaussian of the k^{th} state

n - sample number

N - number of samples in a frame of speech

ω_k - (HMM context) phoneme label

$\omega_t(k)$ - (LSF context) k^{th} LSF coefficient for frame t

Ω - set of M phoneme labels

p - number of coefficients in feature vector

$\mathbf{p}$ - speaker parameter vector

$\boldsymbol{\pi}$ - initial probability vector

π_i - i^{th} coefficient of the initial probability vector

q_t - hidden state for frame t

Q - HMM hidden state sequence

$s(n)$ - speech signal

$s_t(n)$ - speech sample n from frame t

$S_t(m)$ - DFT of $s_t(n)$

σ_s - standard deviation of the speech signal

σ_v - standard deviation of the noise

$\boldsymbol{\Sigma}_{kg}$ - covariance matrix for g^{th} Gaussian of the k^{th} state

t - as subscript frame number

T - as superscript: matrix transpose

$u(n)$ - speech excitation source signal

$v(n)$ - additive noise

$vt(n)$ - vocal tract impulse response

$VT_t(z)$ - vocal tract response for frame t

$w(n)$ - Short-time spectral analysis context: window applied to speech frame

w_k - k^{th} CFD coefficient

$w_t(k)$ - k^{th} CFD coefficient for frame t

$\mathbf{x}_t$ - feature vector for frame t

$x_t(k)$ - k^{th} coefficient of feature vector for frame t

X - observed feature vector sequence

$y(n)$ - noisy speech signal

z - z-domain operator

Chapter 1

INTRODUCTION

The basic principles of Automatic Speech Recognition (ASR) that are in use today have been laid out in the 1970's [1]. Over the years, the advancements in the fields of Digital Signal Processing (DSP), pattern recognition, language modeling, as well as the increase in processing power allowed the creation of real-time continuous, speaker independent (or speaker adaptive), systems with large vocabularies (up to 65,000 words) that are very capable in recognizing clean speech. The recognition rates for such systems with high signal to noise ratio (SNR) input give around 5% error rates [2]. This is a significant achievement considering the size of the vocabulary and the complexity and variability of the speech signals due to differences between speakers, moods, mannerisms, etc. For small vocabulary systems the error rates fall below 1% [2]. In most of the applications however, there is a significant amount of background noise present in the speech signal. Human auditory system, having evolved in environments where the ability to distinguish noise from the signal of interest was a key to survival in many situations, is extremely well adapted for this task. The ASR systems on the other hand have a hard time dealing with situations where there is a relatively small amount of noise present in the speech signal. Being trained under certain conditions and with a finite set of training data, those systems have trouble when the test circumstances are not similar to those at the training time. Needless to say, current state of the art ASR systems cannot match, or come even close, to the human ability to recognize speech in various types of noisy environments. Most of the present methods can deal with a specific type of noise, very often at a cost of decrease in recognition for clean speech. For instance, speech enhancement algorithms typically increase Signal to Noise ratio (SNR) at the cost of intelligibility (and vice-versa).

Research in the field of robust speech recognition has led to a vast array of different approaches and methods, which can be categorized into three classes: speech enhancement, robust feature extraction, and noise independent classifiers. In this work, robustness is defined as ability to cope with noise. Selected methods from all the categories will be presented in the

thesis and used in the simulations for comparison purposes with the focus on robust feature extraction.

1.1 Problem Statement

Most of the applications for speech recognition require the ASR systems to perform in environments with a significant amount of background noise. For instance, a speaker on a busy street is accompanied by noises of passing cars, wind blowing, other people talking in the background, etc. Present ASR technology requires a training of the system, with limited amount of training data, which "teaches" the system how to recognize and categorize certain speech patterns. The background noise, present during the operation stage of the ASR system, often introduces discrepancy between the training and operation conditions, and modifies the patterns of the ASR input. These changes introduce errors in the recognition process which diminish ASR performance.

The human auditory system is an example of speech processing that is immensely robust in presence of all kinds of background noise. This ability is the result of the evolution of certain mechanisms and redundancies in the human speech production and auditory systems for coping with noise that was always present in the surrounding environment. One such robust attribute of the speech is the harmonic structure of the voiced sounds. A good ASR system should exploit the harmonic structure of the speech signal to improve recognition in presence of background noise.

1.2 Objectives

The objective of this thesis is to study current methods that attempt to improve performance of one-microphone, small-vocabulary ASR systems in the presence of additive background noise, and propose a number of feature extraction methods based on the harmonic analysis of the speech signal. This work will deal with white Gaussian noise, a fairly basic type of noise, which can model a large number of background noises encountered in speech processing, and is still problematic for current ASR systems. The thesis will attempt to give an intuitive overview of robust speech processing and present a new method for robust feature extraction. The results of simulations will evaluate the existing and introduced methods.

1.3 Thesis Contributions

This work contributes to the area of Automatic Speech Recognition with the following :

- A new method for data analysis that exploits the harmonic structure of the speech signal and shows more robustness in the presence of additive white Gaussian noise than current methods
- Two variants of the above mentioned method for performance-complexity trade off
- Three feature extraction methods based on the above mentioned method
- A concise and intuitive overview of various branches of ASR technology with consistent notation pertaining to the word-level recognition

1.4 Thesis Outline

The thesis is divided into 6 chapters. Chapter 1 is the introduction. Chapter 2 gives an overview of the current state of the ASR technology. Issues of noise in ASR and some methods developed for coping with it are discussed in Chapter 3. Chapter 4 introduces the new method for speech signal analysis and robust feature extraction, and Chapter 5 gives the results of the simulations that test the new method against existing ones that were presented in previous chapters. Chapter 6 contains the summary and discussion on the results.

There are two Appendices explaining the details of calculation of LPC coefficients and HMM training and classification. The thesis is concluded with the References and Index sections.

Chapter 2

OVERVIEW OF AUTOMATIC SPEECH RECOGNITION

2.1 ASR System Structure

A common ASR system will process a speech input in the following stages [6]:

- Preprocessing – where various operations on the signal are performed in order to prepare it for the feature extraction
- Feature extraction – where the features of interest are extracted from the signal and formed into a set of feature vectors
- Classification – where some type of pattern recognition algorithm assigns labels to feature vectors. Typically this would be done in two stages: training, where the classifier derives the relationship between the feature vector and its proper label, and classification, where the labeling is performed on feature vectors.

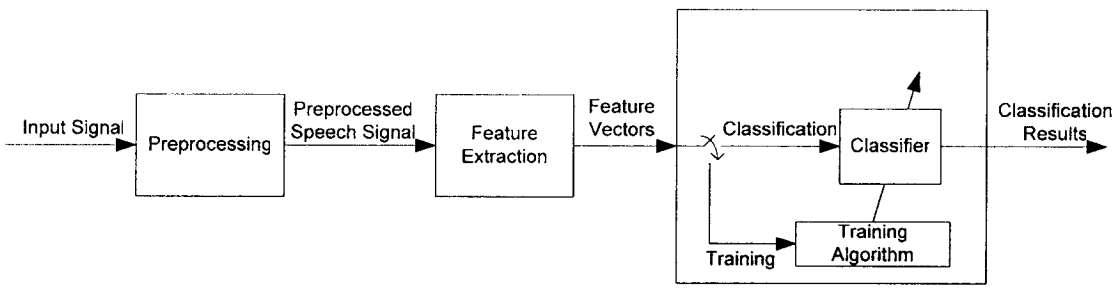


Figure 1 General speech recognition system

It is assumed that speech is already sampled before entering the preprocessing stage. Current ASR systems process speech on a frame basis, deriving a feature vector for each frame and classifying that frame from the feature vector.

2.2 Speech Characteristics and Principles of Auditory Perception

A good overview of speech characteristics and the speech production process can be found in [4],[5],[6] . The speech signal is created by the passage of a sound wave through the human vocal tract. The vocal tract shapes the sound wave into speech with the position of the tongue, lips and operation of vocal cords to produce the desired sounds. This process can be modeled as a nonlinear system acting on a specific excitation source:

$$s(n) = f_v(\mathbf{p}, G_n u(n)) \tag{2.1}$$

where s(n) – is the speech signal

p – the speaker parameter vector, which contains information pertaining to characteristics of individual speakers, such as pitch, length of vocal tract, etc.

u(n) – input, or excitation source

G_n – gain on the input signal

$f_v()$ – nonlinear function representing the vocal tract configuration.

The goal of a speech recognition system is to determine the vocal tract configuration function $f_v()$ from the observation of speech signal s(n). Speaker recognition systems attempt to deduce the speaker parameter vector **p**.

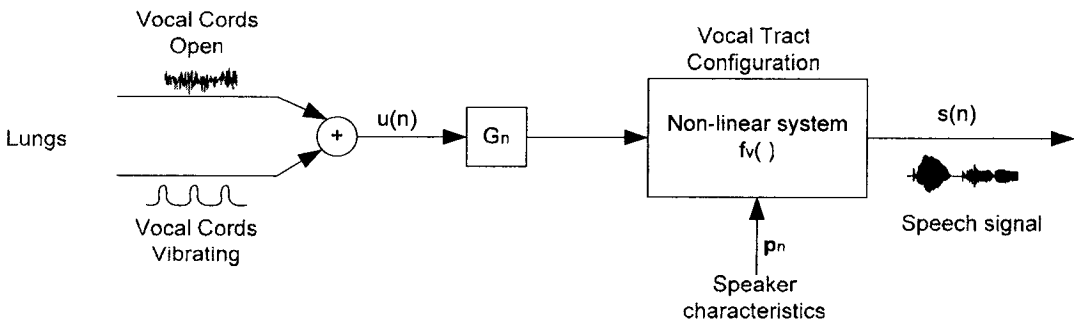


Figure 2 Speech production model

The excitation source u(n) can be generally divided in two categories :

- Noise - the air breathed out when a person speaks
- Periodic impulse - created by vibration of the vocal cords, usually accompanied by a higher gain G_n.

These two types of input result in unvoiced (non-periodic) and voiced (periodic, where the fundamental period is referred to as pitch) sounds, respectively. However, depending on the speaker characteristics and the type of sound produced, the excitation signal might consist of a mix of noise and periodic impulse. The basic acoustic units of speech are called phonemes, and similar phonemes are grouped into families, such as vowels (voiced) and consonants (voiced and unvoiced) [4].

Speech is a non-stationary process, but due to physical limits on how fast the human vocal tract can reconfigure itself, it can be treated as a quasi-stationary process for 10-30ms intervals. This means that it can be treated as a stationary process for those short time periods [4]. The quasi-stationarity assumption is more proper for voiced than unvoiced sounds.

Speech signal can be well characterized in the frequency domain. The frequency span of human speech goes up to approximately 7 kHz [5]. Voiced speech consists of harmonics of the fundamental frequency F0, which is also termed the pitch, and it relates to the fundamental period of the excitation source u(n). The average pitch is around 130Hz and 230Hz for males and females respectively [5], but it varies even for the same speaker, as the tone, expression, or mood of the speaker changes (for instance, a question phrase usually ends on a raised pitch). The resonances of the fundamental frequency amplified by the vocal tract are called formants, and it has been shown that their position on a frequency scale is important for human perception of a given voiced sound [5]. Formants are labeled according to increasing frequency, therefore F1 will be the lowest frequency formant and F5 will be the highest frequency formant*. While for the voiced speech it is the spectral content that is important for

* There are times when the formants, tracked in time, overlap and sometimes switch position, for instance making F3 lower on the frequency scale than F2 [5].

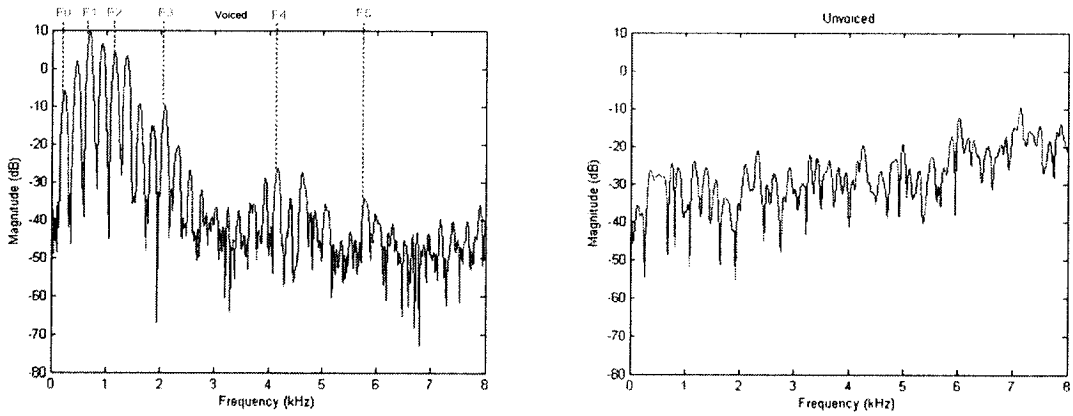


Figure 3 FFT Magnitude of a 20ms frame of spoken phoneme: (a) 'a' – voiced; (b) 's' - unvoiced

perception of given sound, for unvoiced speech it is the changes of the spectral content with time that are of more significance.

The difficulty in speech recognition comes from the diversity of speech sounds which is the result of an enormous number of variables present in the speech production process due to different speaker characteristics, such as pitch, shapes of the vocal tract, length of various parts of the vocal tract, etc. The characteristics of speech will change even for the same speaker, most notably when the mood or expression of the speaker alters. Therefore, for instance in a same type of voiced sound some formants may be less pronounced than others, and in general positions of all the formants vary.

There is a lot of redundant information in the speech signal. Most of the current ASR systems remove the redundancy. For instance, it has been determined experimentally that the human auditory system is less sensitive to phase information than the energy spectrum of the speech signal [6], and therefore almost all the ASR systems today discard the phase information from the speech signal. Experiments have shown that the human auditory system is sensitive to the spectral content of speech on a logarithmic scale – both for the spectrum amplitude and frequency resolution [4],[5].

Current ASR systems use a linear approximation for the nonlinear vocal tract configuration function $f_v(\ )$ in equation (2.1), resulting in a speech model represented as a convolution of a time-variant vocal tract response vt(n), and an input signal u(n).

$$s(n) = vt(n) * G_n u(n) \tag{2.2}$$

This simplification makes the estimation of vt(n) easier at the expense of the response becoming a correlation of the vocal tract configuration and speaker characteristics. The vocal tract is modeled as a tube concatenated from a number of pieces of different lengths and cross-sections, which depend on speaker parameters and the vocal tract configuration [4]. Existing methods attempt to determine the vocal tract response using the knowledge of characteristics of the excitation source u(n), and decorrelate the speaker variability from the speech content.

2.3 Short-time spectral analysis

Since the vocal tract is time-variant, but quasi-stationary, the speech is processed in frames

$$s_t(n) = s(tL + n)w(n) \qquad n = 0,1,..,N-1 \qquad L \le N \tag{2.3}$$

where t – is the frame number

n – the sample number in the frame

N – the total number of samples in a frame

L – the number of new samples that each frame is taking

w(n) in equation (2.3) is a window (a Hamming window for instance) that is used to attenuate the speech signal near the edges of the frame in order to reduce the "leakage" effect in the frequency domain [43]. The transformation of the framed speech into the spectral domain is called the Short-Time Fourier Transform (STFT), and it is based on the Discrete Fourier Transform(DFT), most often implemented with a Fast Fourier Transform (FFT) algorithm

$$S_t(m) = \sum_{n=0}^{N-1} s_t(n) e^{\frac{-j2\pi mn}{N}} \qquad (2.4)$$

$$m = 0, 1, \ldots, N-1$$

where m – is the frequency sample

The FFT algorithm allows for an efficient and real-time computation of the STFT. The spectral content can be visualized with a spectrogram, where the signal power

$$|S_t(m)|^2 = S_t(m)S_t^*(m) \qquad (2.5)$$

for all the frames is shown on a frequency-time axis with a logarithmic scale represented by a color map. Figure 4 shows an example waveform and spectrogram of the speech sequence "dark suit" spoken by a female. This sequence will be used as an example throughout this work, therefore a detailed transcription of the phoneme content is provided. The sequence is 650 ms long sampled at 8kHz.

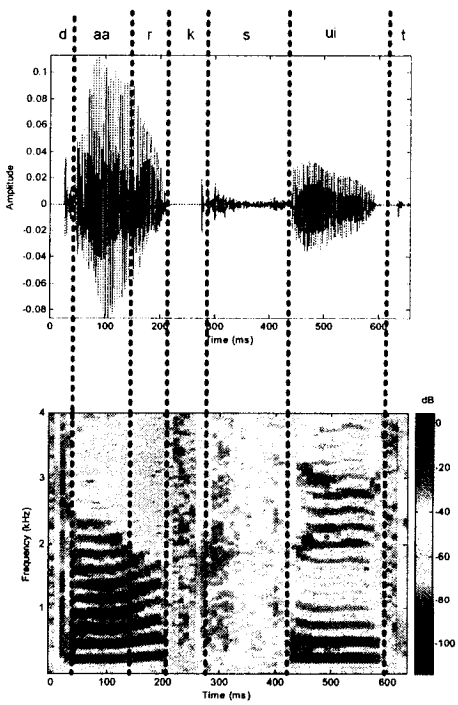


Figure 4 Example of a) speech signal and b) spectrogram for the spoken sequence "dark suit"

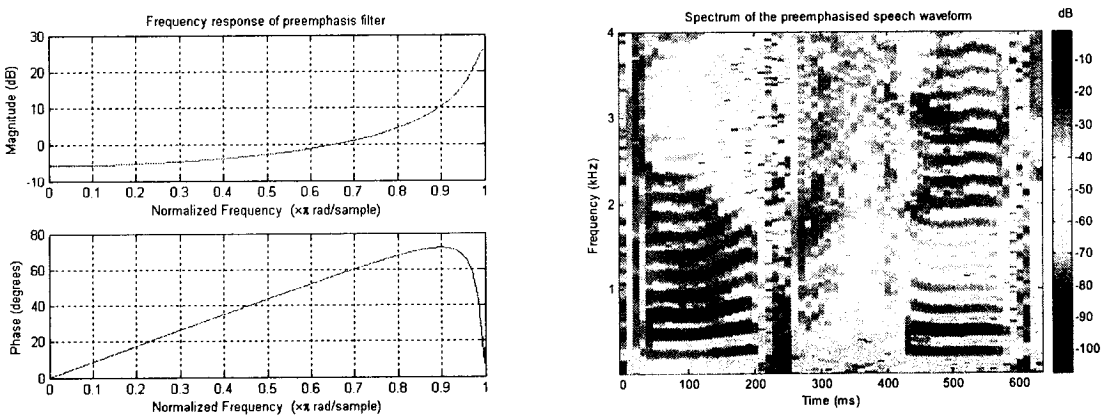
2.4 Preprocessing

In the preprocessing stage of ASR systems various operations on the speech signal are done in preparation for feature extraction. This might involve accentuating characteristics pertaining to speech, or removing the components that are of no interest.

2.4.1 Preemphasis

The preemphasis filter consists of a differentiator used to boost the higher frequencies of the signal, see equation (2.6). This is especially beneficial for voiced sounds, for which the spectral content, on average, falls down at a rate of 6dB/octave - that is the amplitude falls down by 6dB from frequency f to frequency 2f [3]. Although for unvoiced speech this filter tends to decrease the SNR, the benefits of the preemphasis justify the side effects.

$$s(n) = s(n) - as(n-1) \qquad (2.6)$$
$$a \approx 0.9$$



a) b)

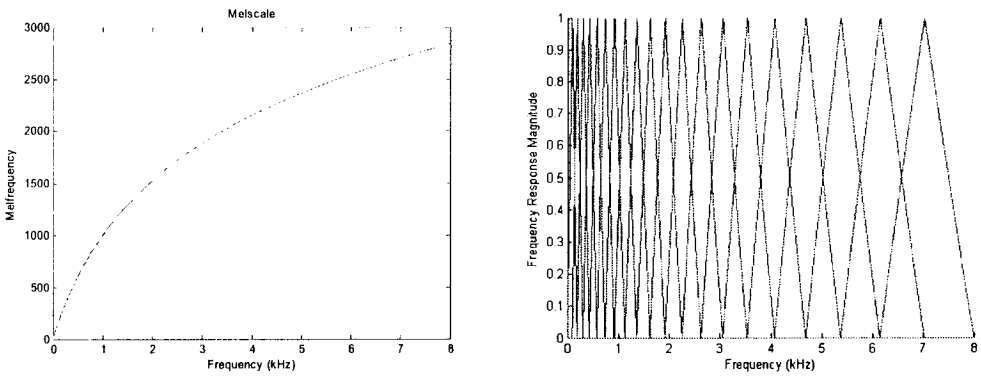
Figure 5 a) Frequency response of a typical preemphasis filter and b) spectrogram of the sequence "dark suit" after filtering with the preemphasis filter

2.4.2 Mel-scale filtering

Mel-scale filtering is an attempt to imitate the human auditory response, which has a higher resolution for low frequency content and a lower resolution for high frequency content [4],[5],[6]. Mel-scale conversion maps the frequencies to a new scale, which is approximately linear up to 1kHz and logarithmic above 1kHz (see Figure 6a).

$$m_{Mel}(f) = 2595\log_{10}(1 + f/700) \tag{2.7}$$

where f is the frequency in Hz.



a) b)

Figure 6 a) Mel-scale; b) 20 filter mel-bank for an 8kHz signal

In Mel-bank filtering, the signal is filtered through a bank of filters created on a mel-scale (see Figure 6b). The output of the filter bank represents a spectrogram mapped to the mel-scale (see Figure 7) and it can be used as the spectrum for later analysis.

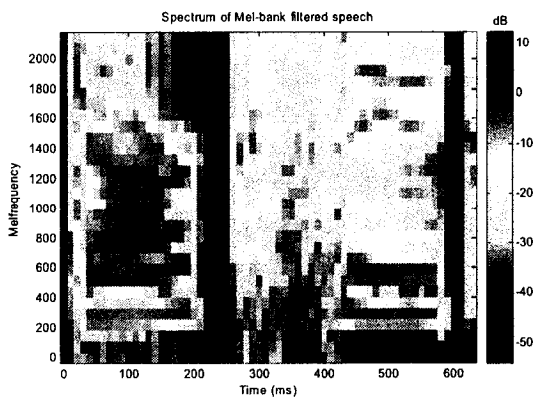


Figure 7 Mel-bank filtered spectrum of the sequence "dark suit"

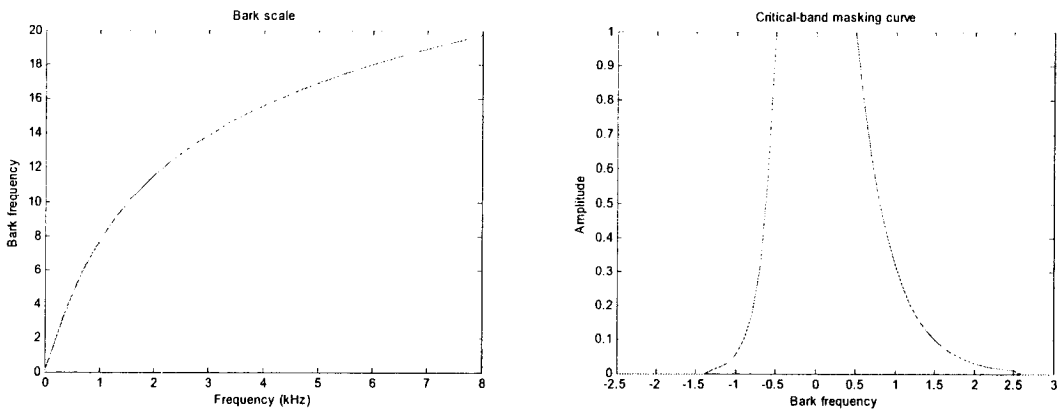
2.4.3 PLP processing

The Perceptual Linear Prediction (PLP) method was introduced in [21] as a feature extraction method that combined perceptual preprocessing of the signal followed by extraction of LPC coefficients. In this work PLP is treated as a preprocessing method only, which can be followed by any feature extraction technique. Like mel-bank filtering, PLP was inspired by characteristics of human auditory system, but it attempts to exploit more characteristics than just the increase of resolution for low frequency content.

The first stage of PLP processing maps the signal to a Bark frequency scale, which is very similar to the Mel-scale (see Figure 8a).

$$m_{Bark}(f) = 6\ln\left(\frac{f}{600} + \sqrt{\left(\frac{f}{600}\right)^2 + 1}\right) \tag{2.8}$$

where f is the frequency in Hz.



a) b)

Figure 8 a) Bark frequency scale; b) Critical-band masking curve

Then, critical band analysis is performed by convolving the Bark scale power spectrum of the signal $P_t(m_{Bark})$ with the critical-band masking curve $\Psi(m_{Bark})$, which approximates the shape of human auditory filters (see Figure 8b).

$$\theta_t(m_{Bark}) = \sum_{k=-1.3}^{2.5} P_t(k - m_{Bark})\Psi(k) \tag{2.9}$$

where $P_t(k) = \mathrm{Re}[S_t(k)]^2 + \mathrm{Im}[S_t(k)]^2$

$$\Psi(m_{Bark}) = \begin{cases} 0 & ,m_{Bark} < -1.3, m_{Bark} > 2.5 \\ 10^{2.5(m_{Bark}+0.5)} & ,-1.3 \le m_{Bark} \le -0.5 \\ 1 & ,-0.5 < m_{Bark} < 0.5 \\ 10^{-1(m_{Bark}-0.5)} & ,0.5 \le m_{Bark} \le 2.5 \end{cases}$$

The increment on k in equation (2.9) is not uniform and it depends on the sampling of $S_t(m_{Bark})$, which becomes denser as m_{Bark} increases. The curve $\theta_t(m_{Bark})$ is re-sampled at approximately 1-Bark intervals and it is preemphasized by a simulated equal loudness curve E(ω)

$$\Xi_t[m_{Bark}(\omega)] = E(\omega)\theta_t[m_{Bark}(\omega)] \tag{2.10}$$

$$E(\omega) = \frac{(\omega^2+56.8\times10^6)\omega^4}{(\omega^2+6.3\times10^6)^2(\omega^2+0.38\times10^9)}$$

where $\omega = 2\pi f$ is the angular frequency in rad/s.

Finally, the signal goes through a cubic-root amplitude compression

$$\Phi_t(m_{Bark}) = \Xi_t[m_{Bark}]^{0.33} \tag{2.11}$$

The PLP processed spectrum can be used for later LPC analysis by taking the Inverse DFT of $\Phi_t(m_{Bark})$, or Cepstrum coefficients can be extracted directly from $\Phi_t(m_{Bark})$ [21].

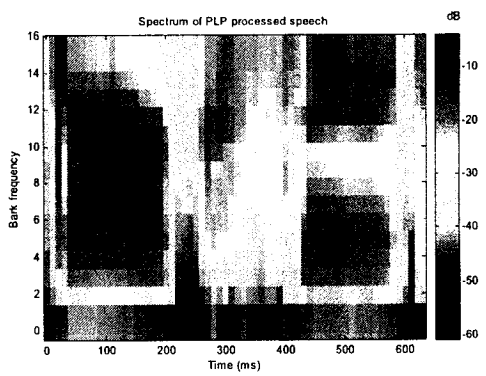


Figure 9 PLP processed spectrum of the sequence "dark suit"

2.5 Feature Extraction

The purpose of feature extraction is to identify key characteristics of the signal that will contribute to recognition. Aside from characterization, this process reduces the dimensionality of the data, which simplifies the classification. The two feature extraction methods presented below are the most common in ASR systems today. Both of those methods model the envelope of the speech frame spectrum in an attempt to extract information pertaining to the vocal tract configuration.

2.5.1 Linear Prediction

Linear Prediction Coding (LPC) is based on the concatenated tube model, where the vocal tract is modeled as a series of tubes of different cross-sections [4]. In the z-domain, the vocal tract is represented as an all pole system

$$V_t(z) = \frac{1}{A(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_t(k) z^{-k}} \tag{2.12}$$

The speech model in equation (2.2) converted to the z-domain gives

$$S_t(z) = VT_t(z) G_{t_n} U_t(z) \tag{2.13}$$

Substituting for $VT_t(z)$ and converting back to the time domain, this gives the Auto-Regressive (AR) speech model, where the current sample is predicted from a linear combination of past samples.

$$s_t(n) = \sum_{k=1}^{p} a_t(k) s_t(n-k) + G_{t_n} u_t(n) \tag{2.14}$$

where p - is the number of coefficients used for the prediction.

$a_t(k)$ – is the k^{th} LPC coefficient of frame t

The LPC coefficients are calculated for each frame and combined into feature vector $\mathbf{x}_t$

$$\mathbf{x}_t = [a_t(1) \quad a_t(2) \quad \ldots \quad a_t(p)]^T \tag{2.15}$$

Typically, p=12 coefficients are extracted, and combined into a feature vector. Each pair of the coefficients normally corresponds to a peak of the spectrum envelope, which for voiced phonemes would correspond to locations of a formant. With the assumption that a speech frame is stationary, the LPC coefficients can be calculated using the autocorrelation function of the frame [5] (see Appendix A for details). This method requires no FFT, but the autocorrelation is still computationally expensive.

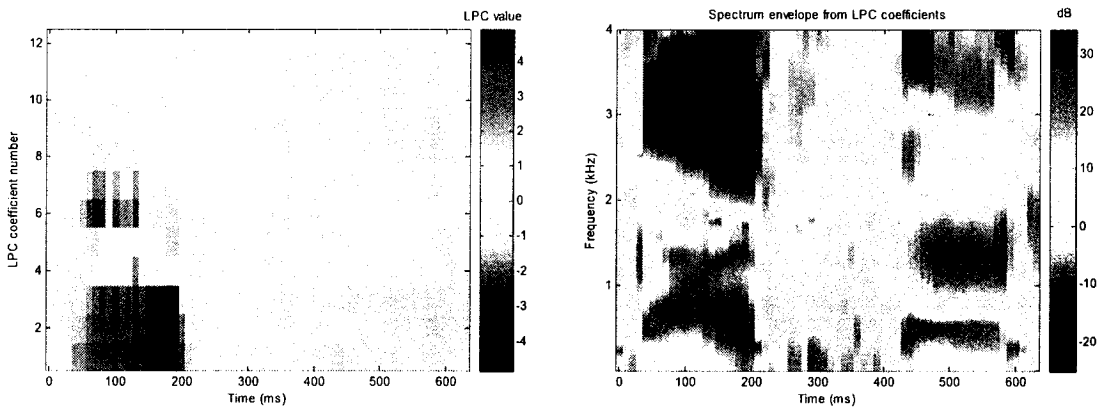


Figure 10. Result of a 12-coefficient LPC analysis and the spectrum envelope they describe for the spoken sequence "dark suit"

2.5.2 Cepstrum Coefficients

Cepstrum coefficients (CC) are another method of modeling the envelope of the frame spectrum [4]. Cepstrum coefficients are calculated by taking the Discrete Cosine Transform (DCT) of the logarithm of the spectrum of the speech frame.

$$c_t(k) = \frac{1}{\sqrt{N}} \log |S_t(0)|^2 + \sum_{m=2}^{N} \sqrt{\frac{2}{N}} \log |S_t(m)|^2 \cos\left(\frac{\pi(2m-1)k}{2N}\right) \tag{2.16}$$

The lower DCT coefficients express the slow variations of the spectrum, ie. the envelope. The higher coefficients correspond to faster variations, such as pitch, which can be used for

speaker recognition/verification (see Figure 11). When the Mel-bank filtered signal is used for the spectrum $|S_t(m)|$, the DCT transformation gives the Mel-frequency Cepstrum Coefficients (MFCC). Typically, 30 Mel-bank filters are used resulting in $|S_t(m)|$ for m=0,1,...29. After the DCT, the lowest 7-12 coefficients are used to form a feature vector – $c_t(0)$ is not used, so the first coefficient is $c_t(1)$.

$$\mathbf{x}_t = [c_t(1) \quad c_t(2) \quad \ldots \quad c_t(p)]^T \tag{2.17}$$

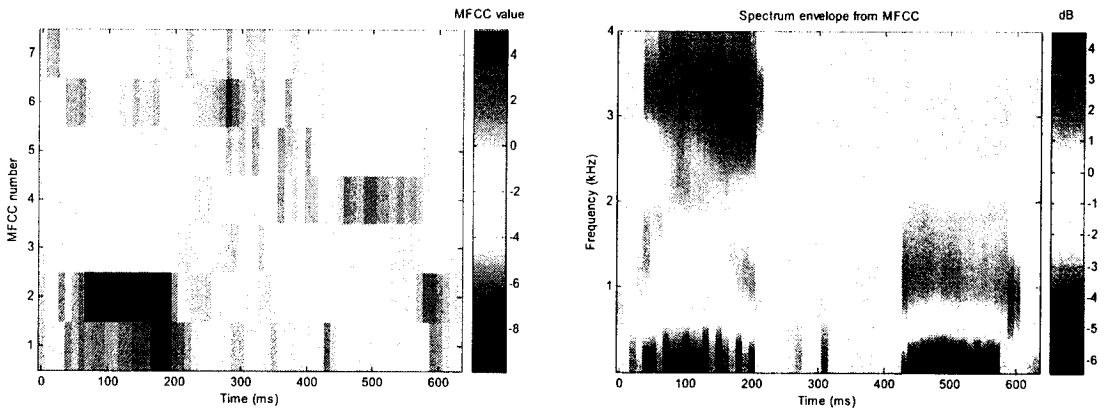


Figure 11 Result of a 7-coefficient MFCC with 20 Mel-bank filters and the spectrum envelope they describe for the spoken sequence "dark suit"

2.6 Pattern Recognition and Classification

The speech model from equation (2.2) is a linear approximation of the true nonlinear speech model. The feature extraction methods discussed above are all based on the linearized approximation, and while they attempt to dissociate what was said from how it was spoken, there still remains a considerable correlation between the speaker and speech content in the feature vectors. Thus, sophisticated classification methods are needed that can deal with complex relations and variability in the feature data. Currently, statistical methods are most widely used to cope with this difficult task, the reason being that varying data can be described or estimated statistically.

General developments and methods of statistical learning theory can be found in [8],[10]. Vapnik [10] indicates three types of statistical learning problems:

- Classification - where the pattern detection is used to categorize data into classes
- Regression - where the patterns are required to estimate unknown continuous signals from noisy observation
- Probability density estimation - where the patterns describe the statistical information about the signal

Speech processing deals with all three problem types: classification for making decisions on speech content in ASR, regression in removing noise from the speech signal, and probability density estimation as an intermediate step in the classification using Hidden Markov Models (HMMs). The $p \times 1$ feature vectors are treated as p-dimensional points in feature space, and depending on the similarity measure used, various patterns in the data can be distinguished. The challenge in statistical pattern recognition is learning complex patterns from a limited set of training data and establishing a proper generalization of the pattern types that does not overfit to particular training set while maintaining the ability to distinguish different classes [10].

These statistical pattern recognition is based on Baye's rule, which constitutes an optimal classifier in the statistical sense [8],[9],[11]. The rule sates that for a $p \times 1$ feature vector $\mathbf{x}_t$, and a set of M class labels $\Omega = \{\omega_1, \omega_2, ..., \omega_M\}$, the posterior probability of a class being labeled ω_k given observed feature vector $\mathbf{x}_t$ is given by the relation

$$P(\omega_k|\mathbf{x}_t) = \frac{p(\mathbf{x}_t|\omega_k)P(\omega_k)}{p(\mathbf{x}_t)} \tag{2.18}$$

The classification is done by finding k, the class label, that maximizes the posterior probability $P(\omega_k|\mathbf{x}_t)$. Since $p(\mathbf{x}_t)$ does not depend on k, it does not contribute to the maximization process, and it can be removed from the equation.

$$label_t = \arg_k \{\max[p(\mathbf{x}_t \mid \omega_k)P(\omega_k)]\} \quad k = 1,...,M \tag{2.19}$$

The prior probability $P(\omega_k)$ is independent of the observed signal, and it is derived from a language model (for instance it could be the probability that a phoneme 'a' is spoken in a K phoneme alphabet) [11]. This thesis is concerned with feature extraction and speech enhancement for small vocabulary systems, where language modeling doesn't play a role as important as in large vocabulary systems, so we remove the prior probability from the equation.

$$label_t = \arg_k \{\max[p(\mathbf{x}_t \mid \omega_k)]\} \qquad k = 1,\ldots,M \tag{2.20}$$

The difficulty in Bayesian classification is that the likelihood probability density function $p(\mathbf{x}_t \mid \omega_k)$, or class-conditional density function, is not known and has to be estimated. The estimation can be done from the training data using Maximum Likelihood (ML) methods. ML algorithms assume a probabilistic model of the data, and find the parameters of the model that give the highest probability of the training data occurring [11].

2.6.1 HMM

In speech, aside from estimating the likelihood $p(\mathbf{x}_t \mid \omega_k)$, it is also important to relate the probabilities for different frames t=1,...,L. This is most often modeled with Hidden Markov Models (HMM). The model attempts to find the hidden state sequence $Q = \{q_1, q_2, \ldots, q_L\}$ from the observed feature vector sequence $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\}$. In a word-recognition problem, the hidden states are related to the phoneme labels $q_t \in \{\omega_1, \ldots, \omega_M\}$, so there are M states for M possible phonemes. The temporal dependencies of the hidden states are assumed to obey the rules of a 1st order Markov process, where:

$$P(q_t = \omega_i | q_{t-1} = \omega_j, q_{t-2} = \omega_k, \ldots) = P(q_t = \omega_i | q_{t-1} = \omega_j) \tag{2.21}$$

These dependencies can be shown unfolded in time by using a Dynamic Bayesian Network (DBN) [32],[33] representation (see Figure 12). The probability of the next state being $q_t = \omega_j$ depends on the previous state $q_{t-1} = \omega_i$ and the likelihood that the feature vector $\mathbf{x}_t$ is observed in state $q_t = \omega_j$.

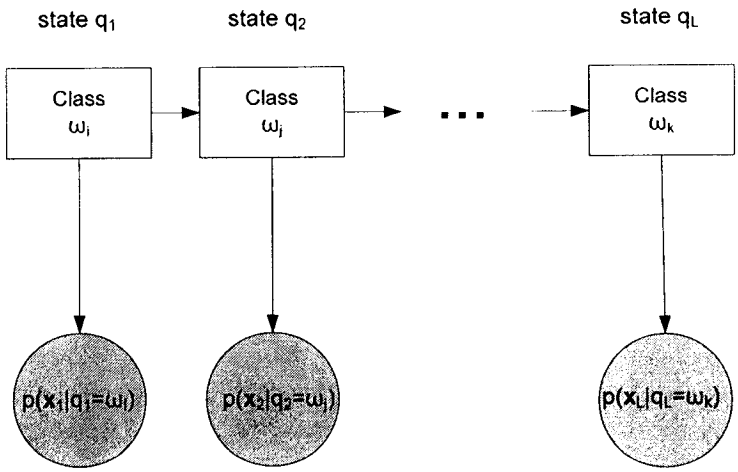


Figure 12 DBN representation of a general HMM model

A complete HMM model $\lambda=(\mathbf{A},\mathbf{B},\pi)$ is described by three parameters:

A – is the state transition matrix, which describes the probabilities of transition from state i to j

$$a_{ij} = P(q_t = \omega_i | q_{t-1} = \omega_j) \tag{2.22}$$

where $1 \le i,j \le M$; $a_{ij} \ge 0, \forall i,j$;

$$\sum_{j=1}^{M} a_{ij} = 1 \quad , \forall i$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1M} \\ . & & & . \\ . & & & . \\ . & & & . \\ a_{M1} & a_{M2} & \dots & a_{MM} \end{bmatrix} \qquad \mathrm{M} \times \mathrm{M}$$

B - the parameters describing the probability distributions for observing a given feature vector at time t, while being in a given state ω_i.

$$b_i(\mathbf{x}_t) = p(\mathbf{x}_t|q_t = \omega_i) \tag{2.23}$$
$$\mathbf{b}_{\mathbf{x}_t} = \begin{bmatrix} b_1(\mathbf{x}_t) & b_2(\mathbf{x}_t) & \dots & b_M(\mathbf{x}_t) \end{bmatrix}^T$$

The estimation of the observation distributions corresponds to estimating the likelihood density in equation (2.20). The observation density is modeled by a continuous distribution function with **B** being the collection of distribution function parameters (see Section 2.6.2 for details).

π – the initial probability vector, which gives the probability of the first state being ω_i

$$\pi_i = p(q_1 = \omega_i) \tag{2.24}$$
$$\boldsymbol{\pi} = [\pi_1 \quad \pi_2 \quad \dots \quad \pi_M]^T \qquad \mathrm{M} \times 1$$

The joint distribution for the whole HMM model is:

$$p(\mathbf{x}_{1:L}, q_{1:L}) = p(q_1)\prod_{t=2}^{L} p(q_t \mid q_{t-1})\prod_{t=1}^{L} p(\mathbf{x}_t \mid q_t) \tag{2.25}$$

The training is done using the Expectation Maximization algorithm, which finds the HMM model that maximizes the likelihood of observation sequence X given a desired state progression (see Appendix B for details).

$$\lambda^* = \arg_\lambda\{\max[p(X|\lambda]\} \tag{2.26}$$

For a small vocabulary ASR, the states in the HMM correspond to different possible phonemes. One HMM is created for each word class, therefore for K words, there are K HMMs $\{\lambda_1,\lambda_2,\dots,\lambda_K\}$. Each HMM is trained with a number of observations from a training set of the corresponding word class (see Figure 13). The classification is done by running a Viterbi algorithm on each HMM to find the best state sequence for a given observation sequence X [11] (see Appendix B for details). The HMM with the highest probability along its best sequence is selected as the word class.

$$label = \arg_k\{\max[p(X \mid \lambda_k)]\} \tag{2.27}$$

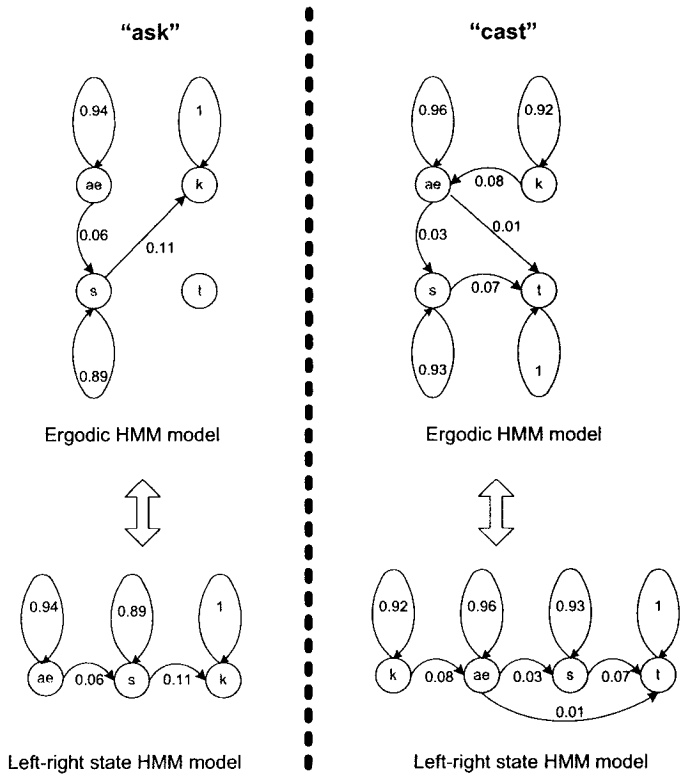


Figure 13. Example of two 4-state HMMs with state transition probabilities for words "ask", and "cast"

2.6.2 GMM

A common model for a class-conditional probability distribution is a Gaussian Mixed Model (GMM). This model is parameterized with G normal distributions $\mathcal{N}_1(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)$, $\mathcal{N}_2(\boldsymbol{\mu}_2,\boldsymbol{\Sigma}_2)$,... $\mathcal{N}_G(\boldsymbol{\mu}_G,\boldsymbol{\Sigma}_G)$ related by mixing coefficients $c_1,c_2,...c_G$. A GMM model is created for each phoneme class ω_k.

$$p(\mathbf{x}_t \mid \omega_k) = \sum_{g=1}^{G} c_{kg}\, p(\mathbf{x}_t \mid \boldsymbol{\mu}_{kg}, \boldsymbol{\Sigma}_{kg}) \tag{2.28}$$

where

$$p(\mathbf{x}_t|\boldsymbol{\mu}_{kg}, \boldsymbol{\Sigma}_{kg}) = \frac{1}{\sqrt{2\pi \det(\boldsymbol{\Sigma}_{kg})}} e^{-\frac{1}{2}(\mathbf{x}_t-\boldsymbol{\mu}_{kg})^T \boldsymbol{\Sigma}_{kg}^{-1}(\mathbf{x}_t-\boldsymbol{\mu}_{kg})}$$

$\boldsymbol{\mu}_{kg}$ – is the p×1 mean vector of the gth distribution for phoneme k
$\boldsymbol{\Sigma}_{kg}$ – is the p×p covariance matrix of the gth distribution for phoneme k

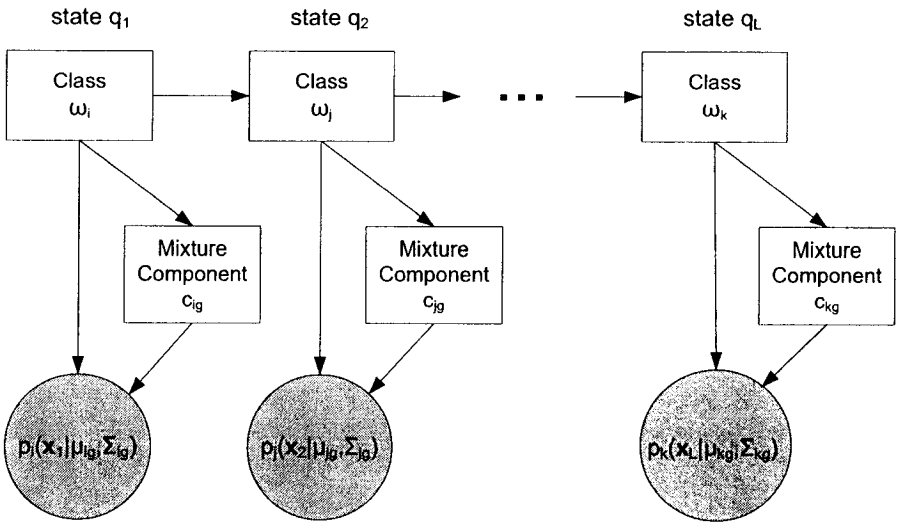


Figure 14 DBN representation of HMM model with GMM observation

The HMM model with GMM class-conditional probability can be represented with a DBN graph (see Figure 14). In HMM notation, the class conditional probability is defined as

$$b_k(\mathbf{x}_t) = p(\mathbf{x}_t \mid \omega_k) = \sum_{g=1}^{G} c_{kg}\, p(\mathbf{x}_t \mid \boldsymbol{\mu}_{kg}, \boldsymbol{\Sigma}_{kg}) = p(\mathbf{x}_t \mid \Theta_k) \tag{2.29}$$

where $\boldsymbol{\Theta}_k = \{c_{k1},\ldots,c_{kG}, \boldsymbol{\mu}_{k1},\ldots,\boldsymbol{\mu}_{kG}, \boldsymbol{\Sigma}_{k1},\ldots,\boldsymbol{\Sigma}_{kG}\}$ is the collection of parameters and mixing coefficients for all G Gaussians corresponding to state k, and $\mathbf{B} = \{\boldsymbol{\Theta}_1,\ldots, \boldsymbol{\Theta}_M\}$ is the collection of all GMMs for all M states. **B** can be derived from supervised training using the EM algorithm (see Appendix B for details).

2.7 Summary

This chapter presented a condensed overview of Automatic Speech Recognition with the most common digital processing and pattern classification techniques in use today. These methods have been shown to be very successful when they operate on a clean speech signal. However in most of the applications some kind of background noise is inevitable. This noise is unpredictable and almost always time varying, although certain assumptions about its characteristics can be made. The next chapter analyses the effects of additive noise on the speech signal and presents methods that attempt to maintain high recognition rates in ASR systems with a lower SNR speech input.

Chapter 3

NOISE IN ASR

Noise contaminates the speech signal and changes the feature vectors. This causes a significant mismatch between the training and test data. Another source of mismatch is the fact that a speaker aware of the noise will often modify his/her speech characteristics in an effort to overcome the noise. This phenomenon is called the Lombard effect [23]. The overall mismatch between the training and testing conditions often increases the error rates even when performing recognition with a higher SNR than in training. A significant amount of research effort has been put into the development of methods for robust ASR and an excellent summary on various methods existing before 1995 has been done in [23]. This work will not attempt a similar overview, but it will briefly present the methods that have been used for comparison purposes in this research, as well as highlight some of the most recent and promising approaches.

3.1 Effects of additive white noise on speech

A general model of noise in speech signal is given by

$$y(n) = s(n) * h(n) + v(n) \tag{3.1}$$

where y(n) – is the observed, or noisy speech signal

s(n) – is the clean speech signal

h(n) – is the convolutional noise

v(n) – is the additive noise

* - represents a convolution operator

In most of the applications the major factor of noise is the additive noise, often with Gaussian characteristics. This work addresses the problems of additive white (Gaussian with zero mean) noise. This simplifies the noise model to:

$$y(n) = s(n) + v(n) \tag{3.2}$$

Over the years, various approaches emerged for noise treatment in speech processing, and they usually correspond to various stages of an ASR system (see Figure 15). Thus:

- Speech Enhancement - these methods attempt to separate the speech content from the noise. Typically these algorithms would be used in a pre-processing stage, to improve the SNR ratio before feature extraction.
- Robust Feature Extraction – these methods, as the name suggests, are employed at the feature extraction stage, and they attempt to find speech features that are not affected by noise
- Noise independent classifiers – these methods employ a special type of classifier that is not affected by changes in features due to certain types of noise.

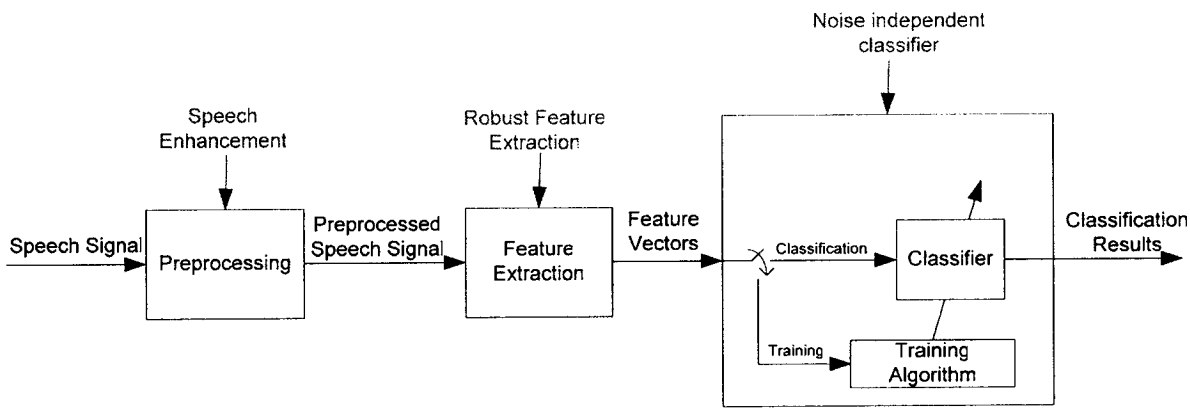


Figure 15 Methods for dealing with noise in ASR systems

Additive white noise has a flat spectrum, meaning it has on average an equal amount of energy in all the frequencies. The assumption that noise is not correlated with speech implies that it is also additive in the frequency domain:

$$Y(m) = S(m) + Q(m) \tag{3.3}$$

The result is a raised mean of the magnitude spectrum of noisy speech that usually drowns the spectral nulls and the weaker frequency components of the signal (see Figure 16).

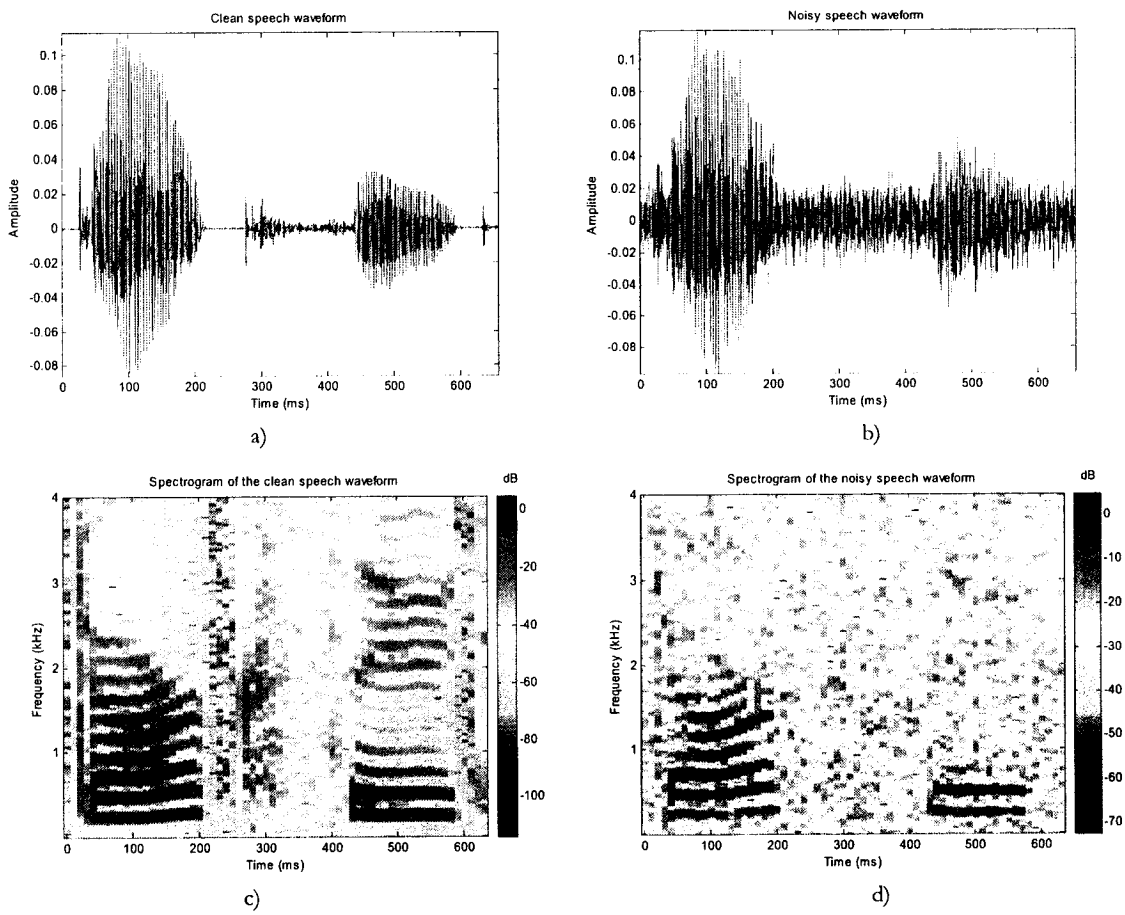


Figure 16 Waveform and spectrum of a clean a), c) and noisy with 5dB SNR b), d) sequence "dark suit"

The human auditory system is well equipped to deal with such noise probably due to:

- The ability to normalize
- The ability to process redundant information and combine it for recognition purposes

3.2 Effects of additive white noise on features

In this section we will examine the effect of noise on the standard features extracted from the LPC and MFCC methods. LPC analysis models the spectral envelope of the signal, and since the noise removes the spectral nulls and flattens out the lower parts of the spectrum, the peaks in the envelope, which are important for recognition, are less pronounced, often disappearing entirely. Differences between weaker and stronger peaks diminish and the resolution is decreased. The peaks tend to be spread out more or less evenly throughout the frequency, rather than concentrate on particular areas of the spectrum, which are more important for recognition (see Figure 17).

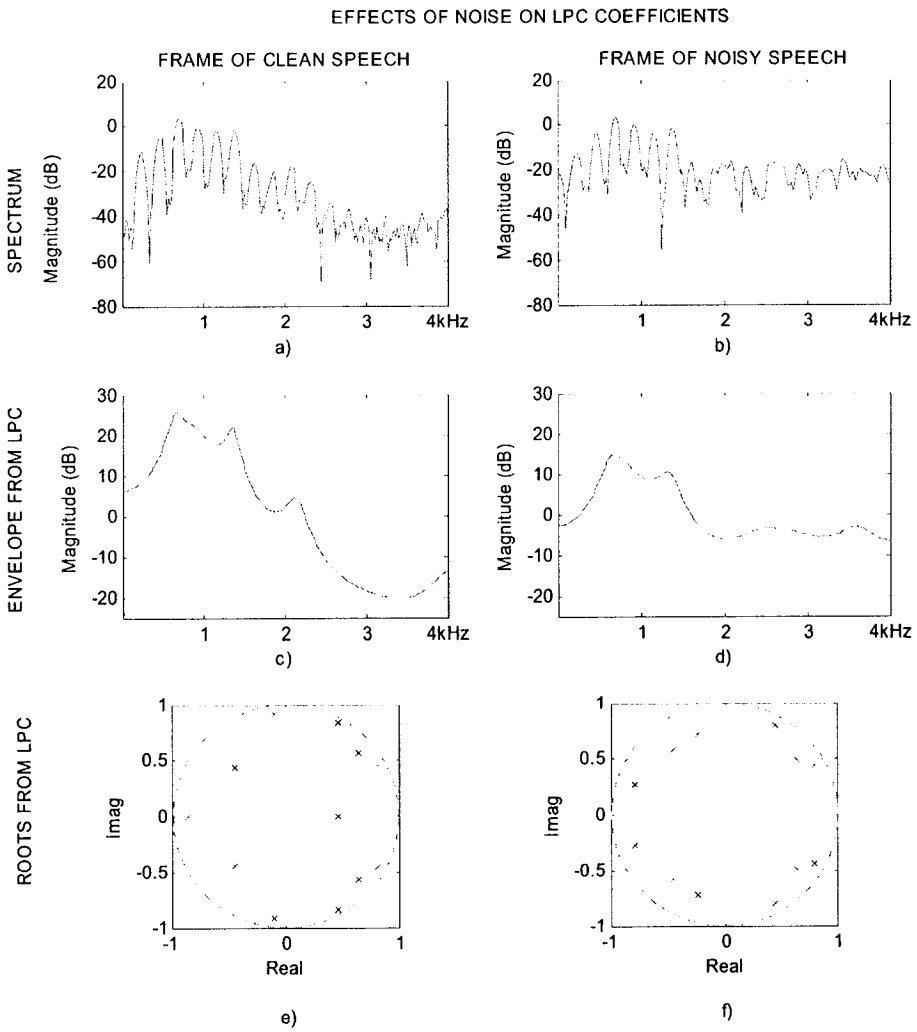


Figure 17 LPC analysis of a frame from the spoken phoneme 'aa' for clean and noisy 5dB SNR signal

MFC coefficients, similarly to LPC, model the shape of the entire spectrum and therefore are affected by the lack of spectral nulls and the flattened envelope. Mel-bank filtering deemphasizes the higher frequency content of the spectrum, making MFCC more robust to noise than LPC for voiced sounds. It has been shown that the norm of the cepstrum coefficients decreases along with the decrease in SNR [23].

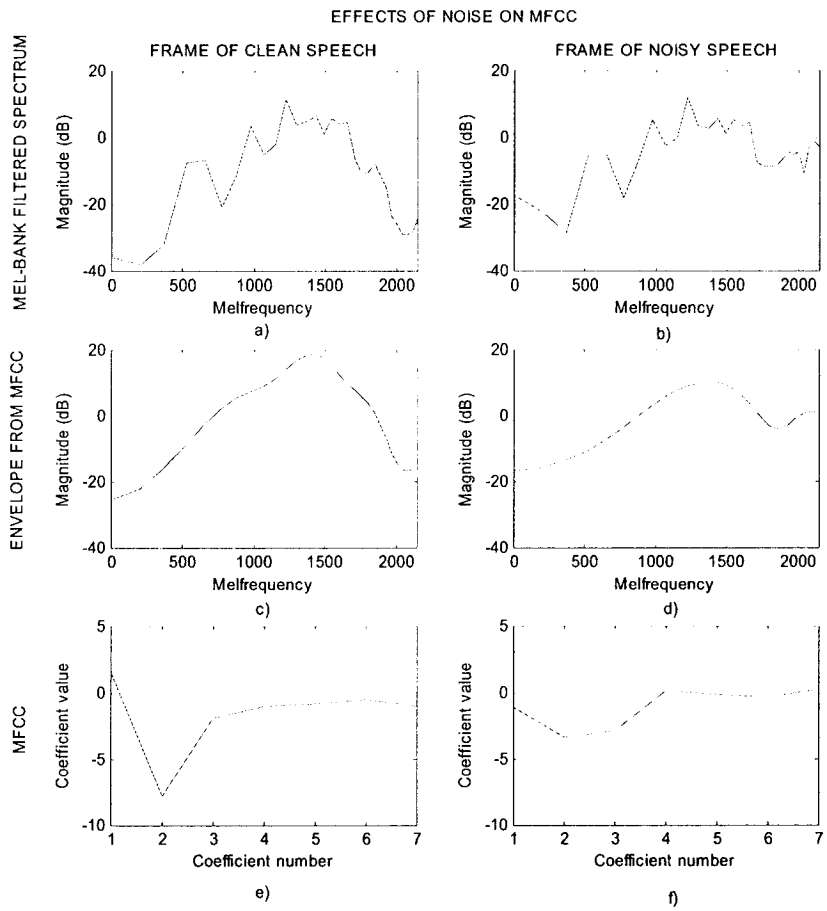


Figure 18 MFCC analysis of a frame from the spoken phoneme 'aa' for clean and noisy 5dB SNR signal

The major disadvantage of LPC and MFCC for dealing with noise is that they represent a snapshot of the entire spectrum envelope of a speech frame in a way that information from various parts of the spectrum is correlated in each coefficient of the representation. This is highly desirable in noiseless situations, because the relationship of various parts of the spectrum captured in the feature vector is important for recognition. However, in the presence of noise this is not desired, as the parts of the spectrum affected by noise which

contain misleading information affect the representation of the parts of the spectrum that have been contaminated to a lesser extent. For instance in LPC, while the position of lower formants in voiced speech, indicated by poles near the right-hand side of Figure 17 e) and f), remains relatively unchanged, the other poles, corresponding to higher formants, differ between the clean and noisy speech signals. Since the LPC coefficients are the results of a polynomial of all the poles, all the coefficients are affected, leaving no part of the feature vector to remain unaltered.

3.3 Speech Enhancement for robust ASR

3.3.1 Kalman Filtering

Kalman filtering (KF) is a very popular and successful method for cleaning the speech content from additive white Gaussian noise [35]. The algorithm usually utilizes the LPC speech model, where the excitation noise of the speech signal and the observation noise are assumed to have Gaussian characteristics. Other models, including non-linear ones, can also be used thanks to Extended Kalman Filtering (EKF) [30]. KF in its initial form required that the variances of the excitation source and the observation noise be known a priori, but new methods have been proposed that do not require any knowledge of noise characteristics and address the issue where the additive noise is colored [34]. Since KF, with the exception of Dual Kalman filtering [30], does not explicitly address the issue of predicting the speech model parameters, but rather relies on other methods to estimate the model parameters in order to perform speech enhancement, it will not be discussed any further in this work.

3.3.2 RASTA

Relative Spectral (RASTA) processing is temporal filtering on each frequency band of the log STFT spectrum of the signal [22]. The assumption here is that the rate of change of speech is limited by the physical structure of the vocal tract, and can be assumed to fit in a certain range. The rate of change of the non-speech components of the signal often will reside outside that range. The RASTA filter is a bandpass filter that suppresses the spectral components of the

signal that have a slower or faster rate of change than that of typical speech. The frequency response of the filter is as follows:

$$H(z) = \frac{0.2z^5+0.1z^4-0.1z^2-0.2z}{z-0.98} \tag{3.4}$$

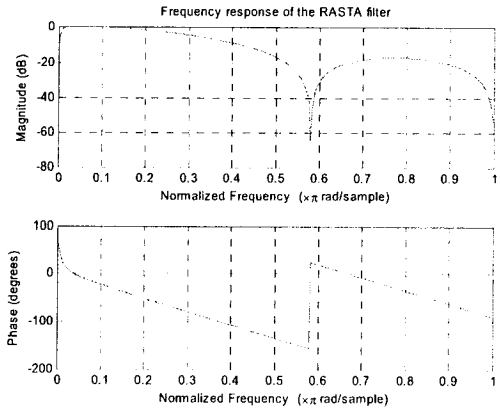
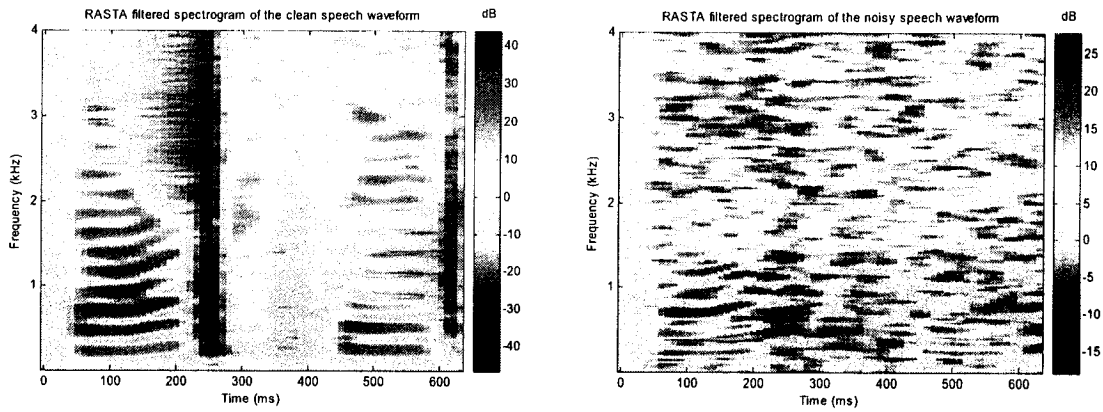


Figure 19 Frequency response of the RASTA filter

The spectrum of RASTA processed clean and 5dB SNR noisy signal is shown in Figure 20.



a) b)

Figure 20 Spectrum of the RASTA processed sequence "dark suit": a) clean waveform; b) 5db SNR waveform

If the signal is pre-processed with PLP, the RASTA filtering is done before cubic-root amplitude compression (see section 2.4.3)

3.4 Robust features

The challenge of deriving robust features is to have features that are unaffected by the noise, and at the same time still convey all the information that is important for recognition. This is not a trivial task, as most of the robust features methods result in decreased performance in noise free conditions.

3.4.1 Line Spectral Frequency

Line Spectral Frequency (LSF) coefficients are a transformation of the LPC coefficients [36],[37]. The inverse filter from equation (2.12) is taken

$$A_t(z) = 1 + a_t(1)z^{-1} + \ldots + a_t(p)z^{-p} \tag{3.5}$$

and used to form two polynomials, from which the roots are calculated to find the LSF coefficients. For even p, the polynomials are calculated as follows:

$$P_t(z) = A_t(z) + z^{-(p+1)}A_t(z^{-1}) = (1 - z^{-1}) \prod_{k=2,4,\ldots,p} (1 - 2z^{-1}\cos\omega_t(k) + z^{-2}) \tag{3.6}$$

$$Q_t(z) = A_t(z) - z^{-(p+1)}A_t(z^{-1}) = (1 + z^{-1}) \prod_{k=1,3,\ldots,p-1} (1 - 2z^{-1}\cos\omega_t(k) + z^{-2})$$

where $\omega_t(k)$ – is the k^{th} LSF coefficient for frame t.

The coefficient order in equation (3.6) is such that $\omega_t(2)<\omega_t(4)<\ldots<\omega_t(p)$ and $\omega_t(1)<\omega_t(3)\ldots<\omega_t(p-1)$. The feature vector is formed as follows:

$$\mathbf{x}_t = \begin{bmatrix} \omega_t(1) & \omega_t(2) & \ldots & \omega_t(p) \end{bmatrix}^T \tag{3.7}$$

It has been shown that LSF coefficients show a better quantization and classification performance than LPC coefficients [38].

3.4.2 Delta features

Delta features attempt to model temporal behavior of the coefficients in feature vectors. They approximates the time derivatives of each coefficient $x_t(i)$ for i=1,2,...p calculated from corresponding coefficient values from previous and following frames t-d,t-d+1,....t+d. Thus, for a feature vector $\mathbf{x}_t$, the delta vector is calculated as follows [29]:

$$\mathbf{x}\Delta_t = \frac{\sum_{k=-d}^{d} k\mathbf{x}_{t+k}}{\sum_{k=-d}^{d} k^2} \tag{3.8}$$

A typical value of d is 3 [29] resulting in a time average over 7 frames. The delta vector $\mathbf{x}\Delta_t$ is appended to the feature vector $\mathbf{x}_t$ forming a new 2p × 1 feature vector.

3.4.3 Cepstral feature adjustements

It has been shown that the norm of the cepstrum coefficients decreases along with the decrease in SNR [23]. A number of normalizing methods have been proposed to remove the effects of noise from the cepstral feature vector. Cepstral Mean Subtraction (CMS) and Cepstral Normalization (CN) normalize the cepstral feature vector with respect to the mean and variance of the vector [24].

$$c_t^{CMS}(k) = c_t(k) - E[c_t(k)] = c_t(k) - \frac{1}{L}\sum_{i=1}^{L} c_i(k) \tag{3.9}$$

$$c_t^{CN}(k) = \frac{c_t(k)}{E[c_t^2(k)]} = \frac{c_t(k)}{\frac{1}{L}\sum_{i=1}^{L} c_i^2(k)} \tag{3.10}$$

Cepstral Gain Normalization (CGN) is a further normalization on the CMS vector [25].

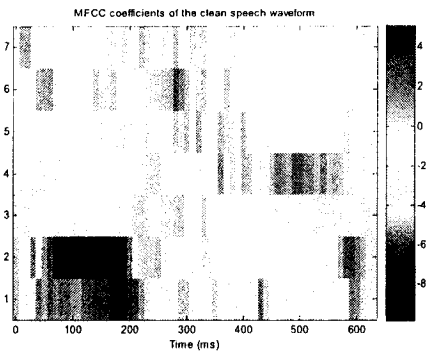
$$c_t^{CGN}(k) = \frac{c_t^{CMS}(k)}{\max_{1\le k\le p}[c_t^{CMS}(k)] - \min_{1\le k\le p}[c_t^{CMS}(k)]} \tag{3.11}$$

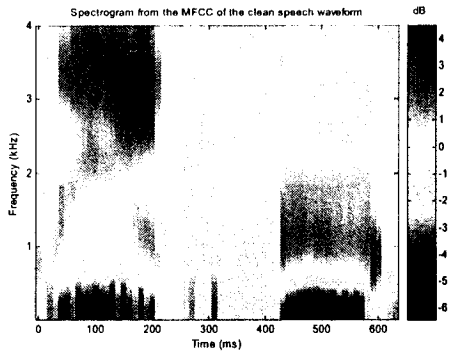
Also, methods have been proposed to use Higher Order Statistics (HOS) to normalize the cepstral feature vectors [26],[27].

7 CEPSTRAL COEFFICIENTS

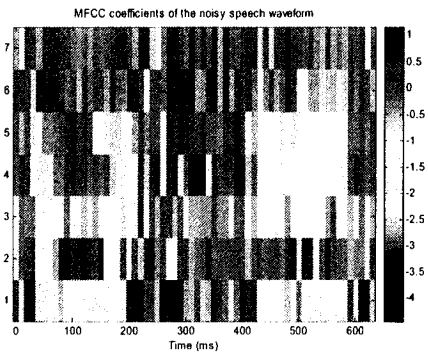
SPECTROGRAM FROM CEPSTRAL COEFFICIENTS

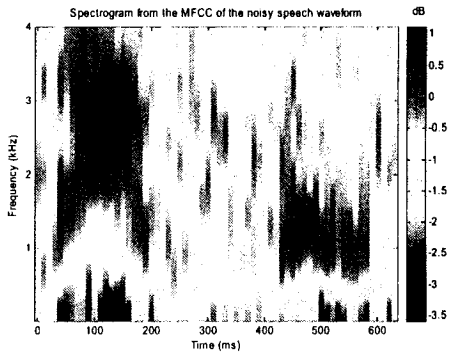
a) Clean Speech MFCC



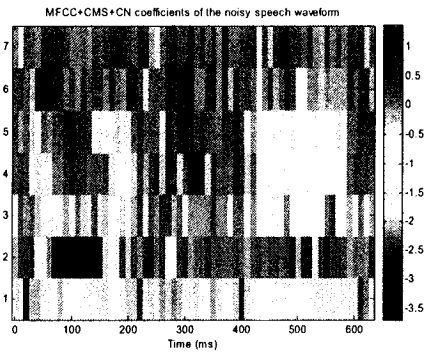


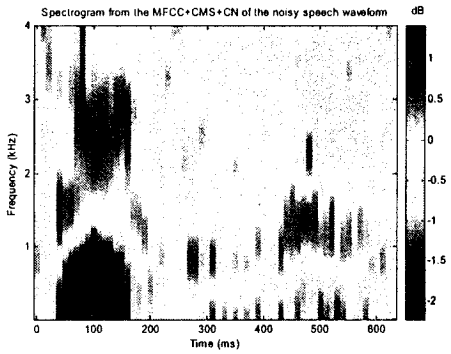
b) Noisy Speech MFCC



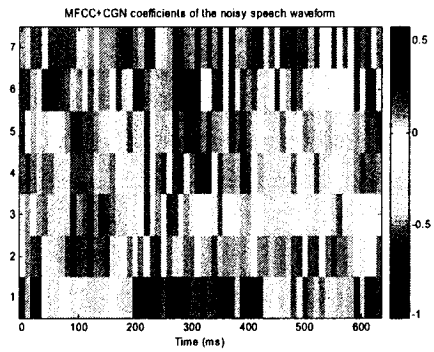


c) Noisy Speech MFCC+CMS+CN





d) Noisy Speech MFCC+CGN



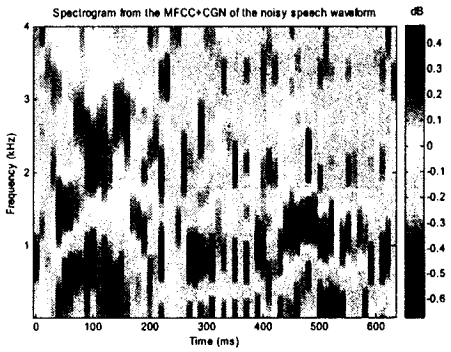


Figure 21 MFCC Coefficients after various normalizations and their corresponding spectrogram envelopes for the speech sequence "dark suit"

3.4.4 Spectral Entropy

A new method for extracting features from the speech signal was proposed in [36]. This method converts the spectrum of the speech frame to a probability mass function (PMF)

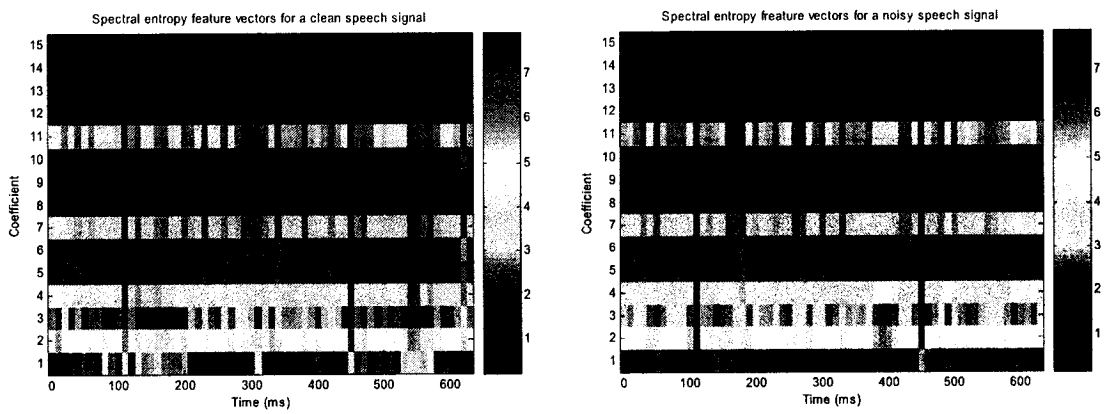
$$S_t^{pmf}(m) = \frac{S_t(m)}{\sum_{i=0}^{N-1} S_t(i)} \qquad m = 0, 1, \ldots, N-1 \tag{3.12}$$

The PMF is then divided into K equal regions, each consisting of $\frac{K}{N}$ samples, and the entropy of each region is calculated according to the formula:

$$E_t^K(k) = -\sum_{i=(k-1)*K}^{k*K-1} S_t^{pmf}(i) \cdot \log_2(S_t^{pmf}(i)) \tag{3.13}$$

This calculation is redone for a varying number of regions, and as suggested in [36] is typically done for K=1,2,...,5. Entropy values from each calculation are concatenated into a feature vector, and so for K=1,2,...,5 this results in a 15 × 1 vector $\mathbf{x}_t$

$$\mathbf{x}_t = \left[E_t^1(1) \quad E_t^2(1) \quad E_t^2(2) \quad E_t^3(1) \quad \ldots \quad E_t^5(4) \quad E_t^5(5)\right] \tag{3.14}$$



a) b)

Figure 22 Spectral Entropy feature vectors for a) clean and b) noisy speech sequence "dark suit"

Spectral entropy features are potentially more resistant to additive white Gaussian noise, because they measure the "peakiness" of various regions of the spectrum [36] - the lower the

entropy, the flatter the region. This makes the measure relatively unaffected by the general increase of mean of the spectrum, although it is still sensitive to the fact that peaks are less pronounced in the noisy spectrum. Also, the regions with peaks that are buried by the noise give a predictable result of entropy going towards zero, rather than producing false peaks as the case may be with LPC or MFCC analysis.

3.5 Noise independent classification

3.5.1 Multi-conditional training

One of the most obvious attempts at removing the discrepancy between training and noisy test conditions is to train a classifier under various scenarios, that is, to do multi-conditional training. For additive white Gaussian noise, the training can be repeated with the same training data at various SNRs, thus enabling the classifier to adapt to features derived from corrupted signal. The problem with this approach is that only a final set of predetermined noise scenarios may be used for training, which may still differ considerably from the noise during testing. Also, the noisy training data may create enough overlap between the features from different classes, that it is not possible anymore to obtain a good generalization of the pattern classes.

3.5.2 Missing Data approach

An interesting and very promising concept for robust ASR and speech enhancement are the algorithms that are generally classified as the Missing Data approach [40],[41]. These algorithms attempt to identify certain regions of the speech spectrum that are less affected by the noise. From these a spectrographic mask is formed [40], which separates spectrum into "reliable" and "unreliable" regions. For ASR the training is done with the information contained in the "reliable" part of the spectrum, and similarly the recognition is done using only the "reliable" portion of the data. A simplest example for voiced signals would be using only the low frequency content of the spectrum for training and recognition, because it is less affected by noise. In order to enhance speech, a relationship must be established between "reliable" and "unreliable" characteristics, which later can be used to recover the missing data of the noisy speech. The performance of these algorithms depends on the ability to form

good spectrographic masks, which will correctly indicate the "reliable" data, while retaining enough of the spectrum to allow for proper recognition.

3.5.3 Hybrid HMM classifiers

In Chapter 2, an HMM with GMM for observation probability density function was presented as a common classifier used in ASR. GMM however is not the only, and not necessarily the best, model for observation probabilities. The hybrid HMM classifiers proposed in [16],[17] combines the ability of HMM to learn temporal patterns from feature vectors with the power of Artificial Neural Networks (ANNs) [18] to form complex non-linear mapping of feature vectors to HMM states. In this approach, Multilayer Perceptron Networks (MLPs) and Recursive Neural Networks (RNNs) are trained to produce probabilities of observing feature vector $\mathbf{x}_t$ while being in state $\mathbf{q}_t$. The ANN has p inputs corresponding to the number of coefficients of a p × 1 feature vector, and M outputs corresponding to the states of the HMM $q_t \in \{\omega_1, \ldots, \omega_M\}$. The number of hidden states and neurons is arbitrary, but good suggestions based on experimental data are given in [16], and [17]. When presented with a feature vector $\mathbf{x}_t$ at the input, the output gives the set of probabilities $\mathbf{b}_{\mathbf{x}_t}$ which are used in the HMM classification (see 2.6.1 and Appendix B for details on HMM training and classification).

Temporal characteristics of the data can be included in the probability derivation by concatenating a number of feature vectors $\mathbf{x}_t$ from different frames at the input of the MLP, or by using RNNs which implicitly capture temporal characteristics of the input data. The advantage of ANNs over a GMM model is that they can capture much more complex dependencies in the data and may be more robust to noise. On the other hand this comes at the price of a much longer and intense training, and the danger of overfitting the classification to the training data. Support Vector Machines (SMVs) [20] have been suggested as an alternative classifier to ANNs for Hybrid HMM approaches [19].

3.6 Summary

In this chapter, the effect of additive noise on speech has been discussed and a number of existing methods for coping with noise in ASR systems has been presented. The next chapter will introduce the proposed new robust feature extraction method based on Comb Filter Decomposition.

Chapter 4

A NEW ROBUST FEATURE EXTRACTION METHOD BASED ON COMB FILTER DECOMPOSITION

The Comb Filter Decomposition (CFD) is an attempt at analyzing the speech signal in terms of its harmonic contents. The motivation is to emphasize the harmonic peaks in the speech spectrum from which we can obtain feature vectors that are less sensitive to noise, especially for voiced signals. This chapter gives a short introduction to comb filters and presents the Comb Filter Decomposition analysis of the speech signal.

4.1 Feedback Comb Filter

A comb filter is a filter with a frequency response consisting of a number of peaks spread equally throughout the frequency axis. A basic comb filter is a k-delay feedback filter with transfer function:

$$H_k(z) = \frac{1}{1 - w_k z^{-k}} \tag{4.1}$$

This filter has a graphical representation shown in Figure 23: z^{-k} in the diagram represents the delay with k being the number of samples the signal is delayed by; w_k is the coefficient of the delayed sample, u(n) is the input and s(n) is the ouput.

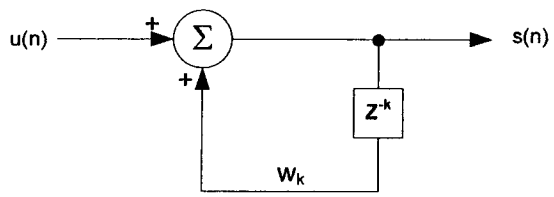


Figure 23 Feedback Comb Filter

The input-output relationship of the feedback comb filter is:

$$s(n) = u(n) + w_k s(n-k) \tag{4.2}$$

The delay k determines the number of harmonics – there are k harmonics in the frequency range [-π,π] of the filter. This characteristic of the comb filter becomes apparent when looking at the location of the poles and zeros of the filter's transfer function in the z-domain. There are k zeros at 0, and k regularly spread equal-magnitude poles z_τ, where τ=1,..,k.

$$H(z) = \frac{1}{1-w_k z^{-k}} = \frac{z^k}{z^k - w_k} \tag{4.3}$$

for $w_k > 0$: $z_\tau = w_k^{\frac{1}{k}} e^{j\tau\frac{2\pi}{k}}$

for $w_k < 0$: $z_\tau = w_k^{\frac{1}{k}} e^{j\tau\frac{2\pi}{k}} = |w_k|^{\frac{1}{k}} e^{j(\tau+\frac{1}{2})\frac{2\pi}{k}}$

where $\tau = 1, \ldots, k$

The coefficient w_k is real, its sign determines the offset of the poles from the real axis, and its absolute value dictates their magnitude (see Figure 24).

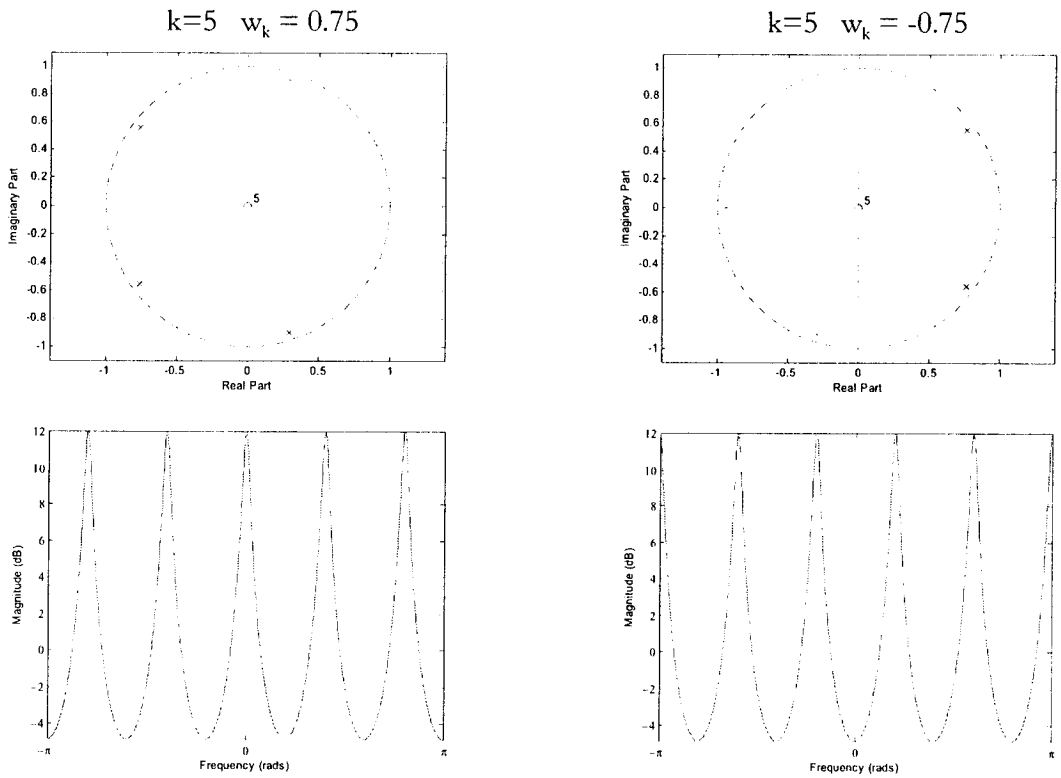


Figure 24 Zero-pole representation and magnitude of the frequency response of k=5 comb filter with a) w_k=0.75; b) w_k= -0.75

Figure 25 shows the magnitude of the frequency response of comb filters for varying values of k and w_k.

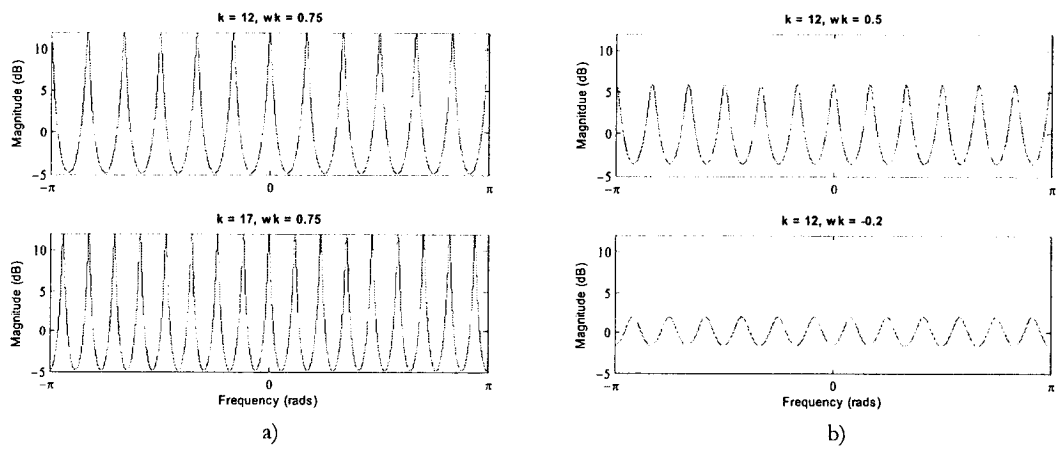


Figure 25 Frequency response of feedback comb filter for a) varying k and fixed w_k; b) fixed k and varying w_k

4.2 Comb Filter Decomposition

In the proposed Comb Filter Decomposition (CFD) the speech spectrum is modeled by a series of independent comb filters of varying delays. The goal is to determine the values of the coefficients w_k for various delays that give the best fit of the speech spectrum. This can be accomplished by rewriting equation (4.2) to have

$$u(n) = s(n) - w_k s(n-k) \tag{4.4}$$

and finding w_k that minimizes u(n) for a given k. This value can be found from the least mean squares method [42] . The cost function J is defined as the expectation of $u^2(n)$, which from equation (4.4) results in:

$$J = E[u^2(n)] = E[s^2(n)] - 2w_k E[s(n)s(n-k)] + w_k^2 E[s^2(n-k)] \tag{4.5}$$

Then the derivative of J with respect to w_k is taken

$$\frac{dJ}{dw_k} = -2E[s(n)s(n-k)] + 2w_k E[s^2(n-k)] \tag{4.6}$$

Setting the derivative $\frac{dJ}{dw_k}$ to zero, equation (4.6) can be solved for w_k

$$w_k = \frac{E[s(n)s(n-k)]}{E[s^2(n-k)]} = \frac{r_s(-k)}{r_{s-k}(0)} \tag{4.7}$$

with $$r_s(-k) = \sum_{n=0}^{N-1} s(n)s(n-k)$$

$$r_{s-k}(0) = \sum_{n=0}^{N-1} s(n-k)^2$$

The expectation of s(n)s(n-k) is the autocorrelation of s(n) at lag –k and the expectation of s(n-k)s(n-k) is the autocorrelation of s(n-k) at lag 0. The quasi-stationary assumption of the speech signal is not used here.

The CFD finds the value of w_k for k=1,2,3,...K using equation (4.7). This gives the coefficients of different comb filters, each producing a different set of harmonics that approximate the signal.

Figure 26 shows the diagram of the CFD method as compared to the LPC. In essence, the CFD filter is a structure that finds different sets of harmonics in the signal independently, rather than finding the shape of the spectrum envelope.

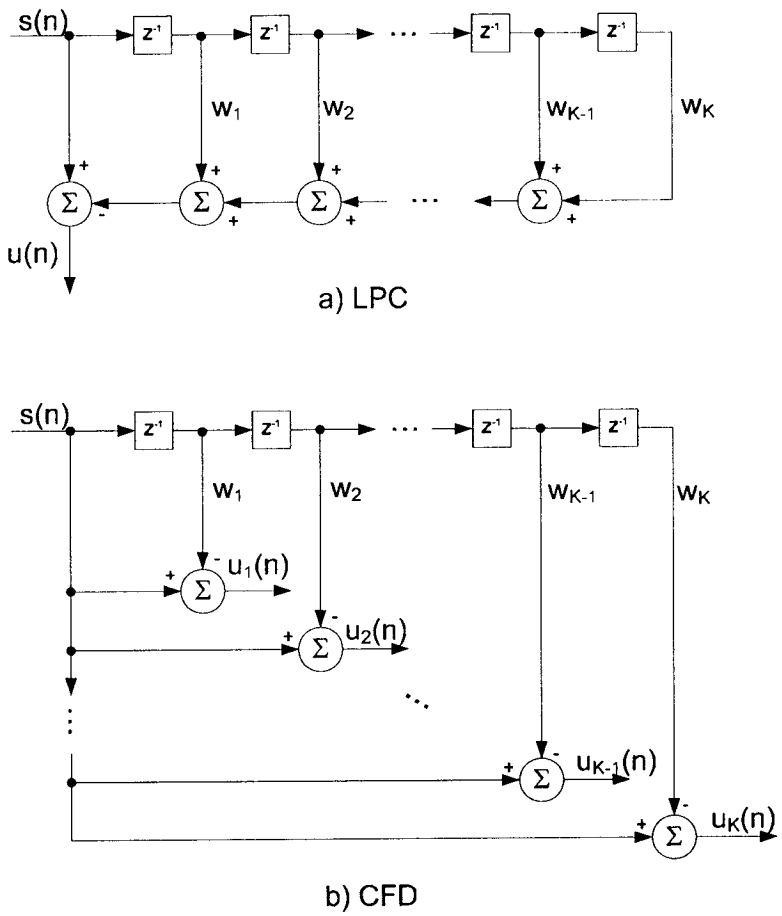


Figure 26 Diagram of a) standard LPC and b) CFD filter

An example of a CFD coefficient vector for up to 160 lags calculated from a single frame of speech is shown in Figure 27 a). Figure 27 b) and c) show how the comb filters for selected lags approximate the spectrum of the speech sound.

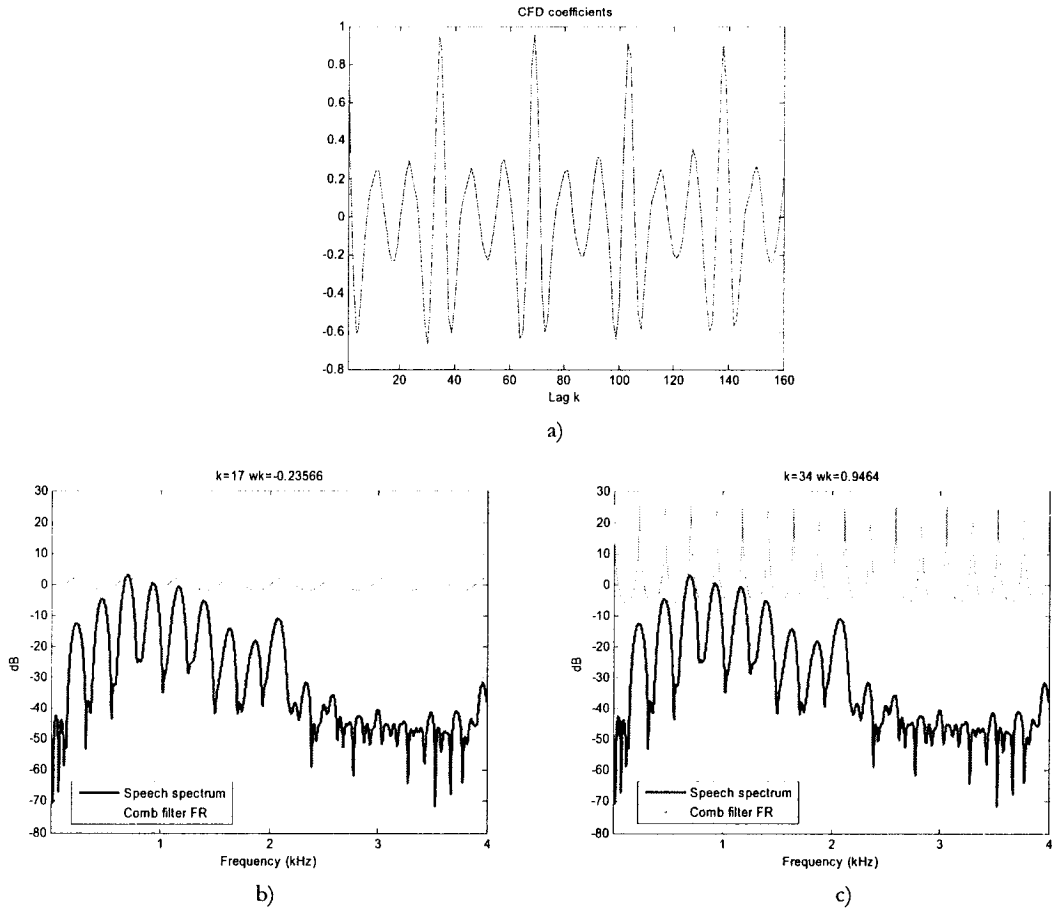


Figure 27 a) CFD coefficients values for frame of phoneme 'aa'; Magnitude of the spectrum of the frame with the magnitude of the comb filter frequency response for the b) 17th, c) 34th lag coefficient

If we assume that the speech signal s(n) is stationary during the time period K, we can replace $r_{s\text{-}k}(0)$ in equation (4.7) with $r_s(0)$ and the CFD becomes the autocorrelation sequence normalized by the autocorrelation value at lag zero. This simplifies the computational complexity of the method, but the stationarity assumption might not be valid for rapidly changing speech. This normalized autocorrelation will be referred to as ACFD.

Equation (4.7) is similar to the definition of the autocorrelation detector (COR) in [44] and [45]. The Subband-Autocorrelation (SUBCOR) method described in those papers is different from the proposed CFD method in the fact that it is a filter-bank analysis stemming from a joint synchrony/mean-rate model of speech processing [46], while the CFD is the calculation

of the coefficients that give, independently and in the mean squares sense, the best fit of different comb filters to the speech spectrum.

The CFD analysis is somewhat similar to the LMS harmonic canceller in [47], where the frequency response of a comb filter is also modeled to fit the input signal's spectrum. Although the intent of the canceller is to remove an undesired component of a known fundamental frequency, the weights of this adaptive filter could be potentially used as feature vectors, which would model the envelope of the speech signal at a known pitch interval. The difference is that the CFD requires no knowledge of the fundamental frequency, and the set of derived coefficients represents the weights for a number of single-weight comb filters (of various harmonics), as opposed to one comb filter of a specific harmonic (as it is in [47]). The CFD is designed to produce a feature vector with coefficients corresponding to different harmonics in the signal frame, while the LMS harmonic canceller would produce a feature vector per each harmonic.

4.3 Features from CFD

In this section three methods will be proposed for deriving features from the CFD (these methods apply to the ACFD as well). The simplest one is forming a feature vector from the first 12 CFD coefficient for delays k=1,...12.

$$\mathbf{x}_t = [w_t(1) \quad w_t(2) \quad \ldots \quad w_t(12)]^T \tag{4.8}$$

These coefficients correspond to the first 12 lowest harmonic comb-filters, where the lower harmonics are those that have a smaller number of harmonic peaks in the spectrum. These low harmonics describe the general shape of the speech spectrum, which is related to the envelope of the true spectrum.

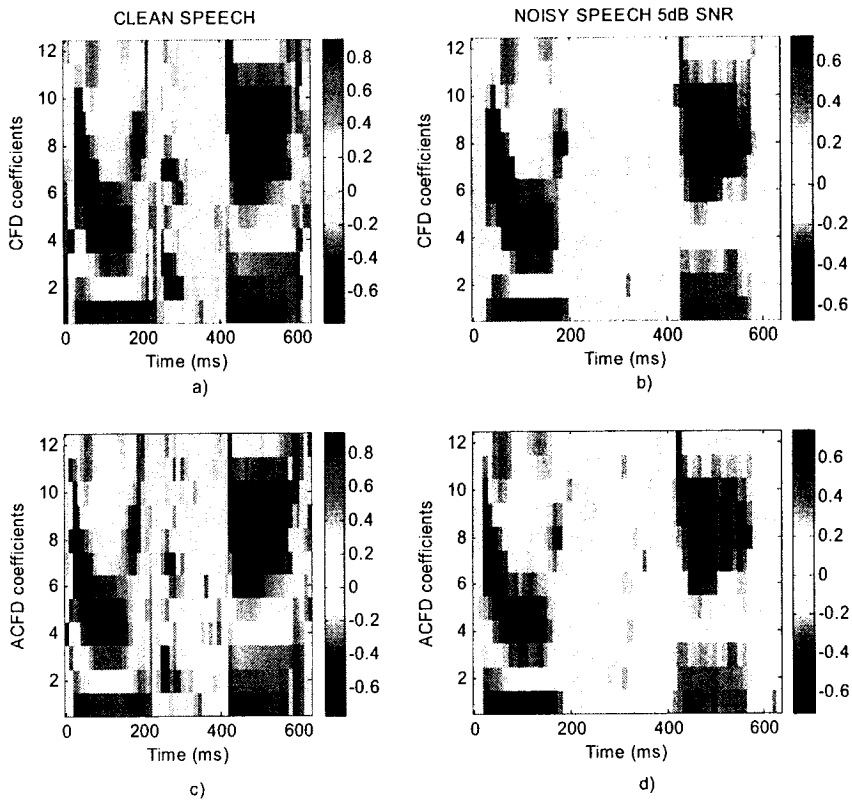


Figure 28 12 CFD and ACFD coefficients for the spoken sequence "dark suit" in clean and noisy conditions.

Another proposed feature extraction method is calculating the cascade frequency response of the comb filters, and then computing of a LPC analysis on the inverse FFT of the resulting cascade spectrum. First, the frequency response $\hat{H}_k(m)$ for each comb filter k=1,...,K is calculated by substituting $z = e^{+j2\pi\frac{m}{K}}$ in equation (4.1) for $m = 0,1,...K-1$

$$\hat{H}_k(m) = H_k(e^{+j2\pi\frac{m}{K}}) \tag{4.9}$$

Then the magnitudes of the frequency response of all the comb filters are multiplied to form the CFD cascade spectrum. Next, the natural log is taken to compensate for the exaggeration of the harmonic peaks due to the previous multiplication. The cascade spectrum is divided by K for normalization:

$$\left|H(m)\right| = \frac{1}{K} \ln \prod_{k=1}^{K} \left|\hat{H}_k(m)\right| \tag{4.10}$$

Figure 29 shows an example of the true spectrum and the CFD and ACFD cascade spectra of the frame for the spoken phoneme 'aa' under clean and noisy conditions. It can be observed that for voiced signals the CFD and ACFD cascade spectrums are fairly immune to additive white Gaussian noise. This is because the CFD (and ACFD) cascade spectrum emphasizes the harmonic peaks of the speech spectrum, which are less susceptible to corruption by additive white noise than other parts of the speech spectrum. From $\left|H(m)\right|$, an inverse FFT* is taken and a standard LPC analysis is done to derive a feature vector from the envelope of $\left|H(m)\right|$. Figure 30 demonstrates the robustness of LPC coefficients derived from CFD and ACFD cascade spectra as compared to standard LPC analysis for the voiced phoneme 'aa'.

* Some re-ordering may be needed to compensate for the wrap-around effect of the inverse FFT before doing the LPC analysis.

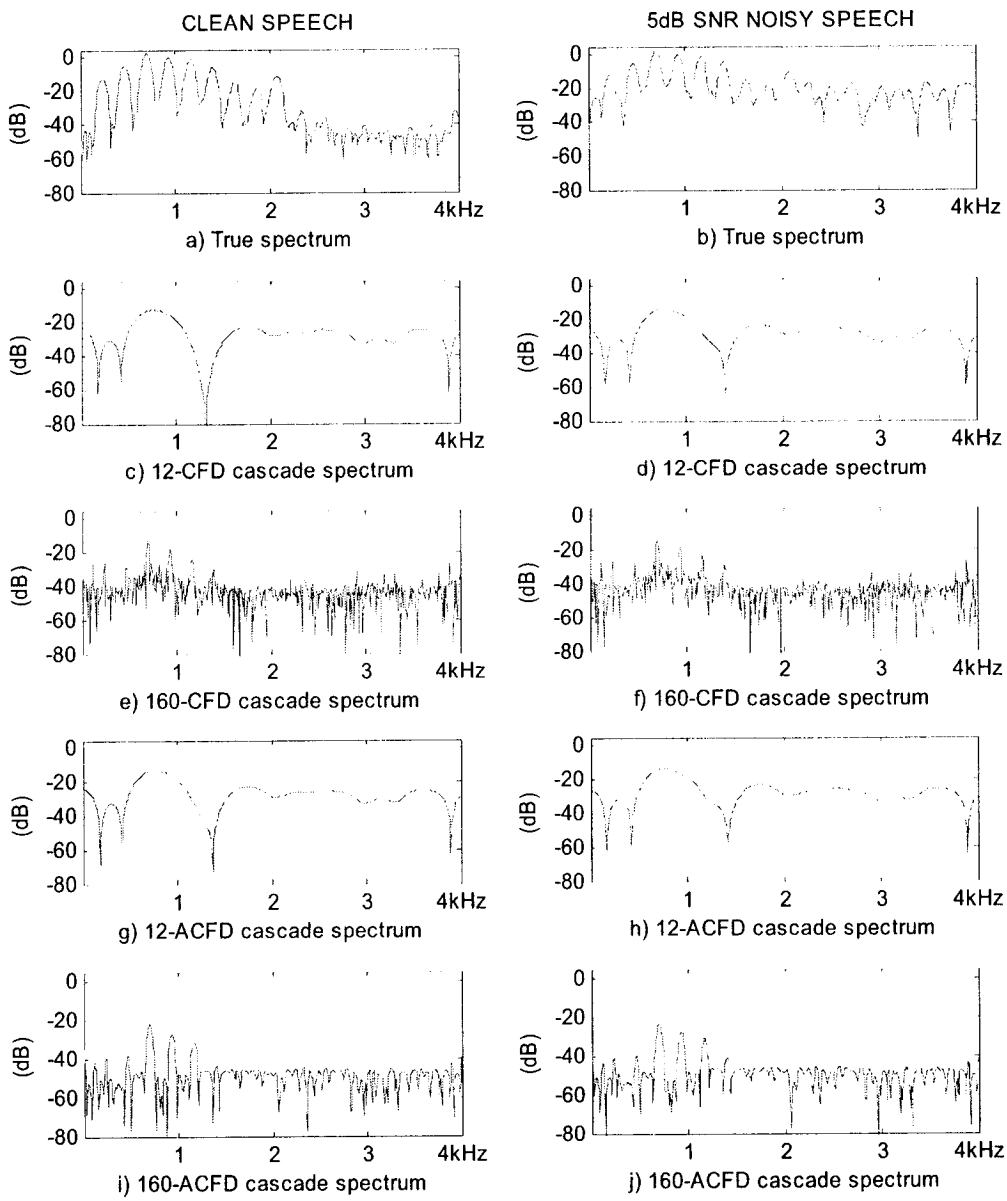


Figure 29 Magnitude speech spectrum of a), b) clean and noisy speech frame for phoneme 'aa'; c), d) corresponding 12-CFD cascade spectrum; e), f) corresponding 160-CFD cascade spectrum; g), h) corresponding 12-ACFD cascade spectrum; i), j) corresponding 160-ACFD cascade spectrum

a)

b)

c)

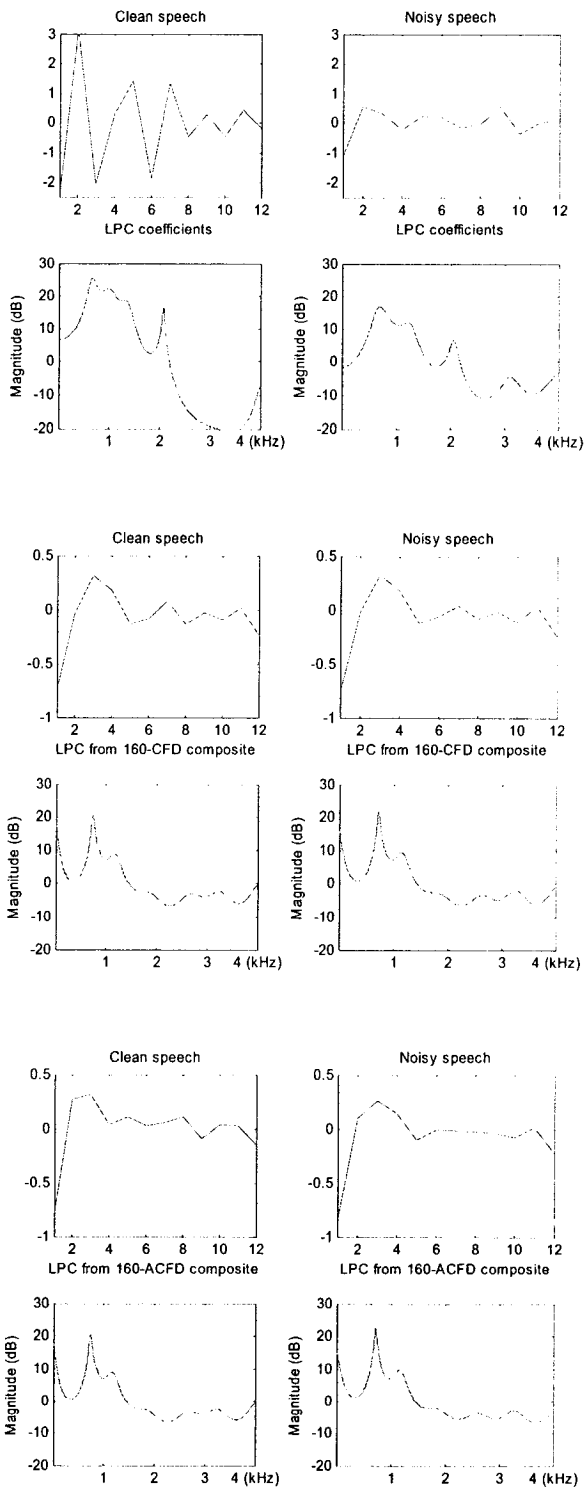


Figure 30 LPC coefficients and their corresponding spectral envelopes from a) speech signal; b) 160-CFD cascade; c) 160-ACFD cascade for clean and 5dB SRN noisy frame of phoneme 'aa'.

The third feature extraction method follows the same steps as the previous method with the addition of LSF conversion after the LPC analysis. Figure 31 shows an example of LSF coefficients formed from the CFD and ACFD cascade spectrum for the spoken sequence "dark suit".

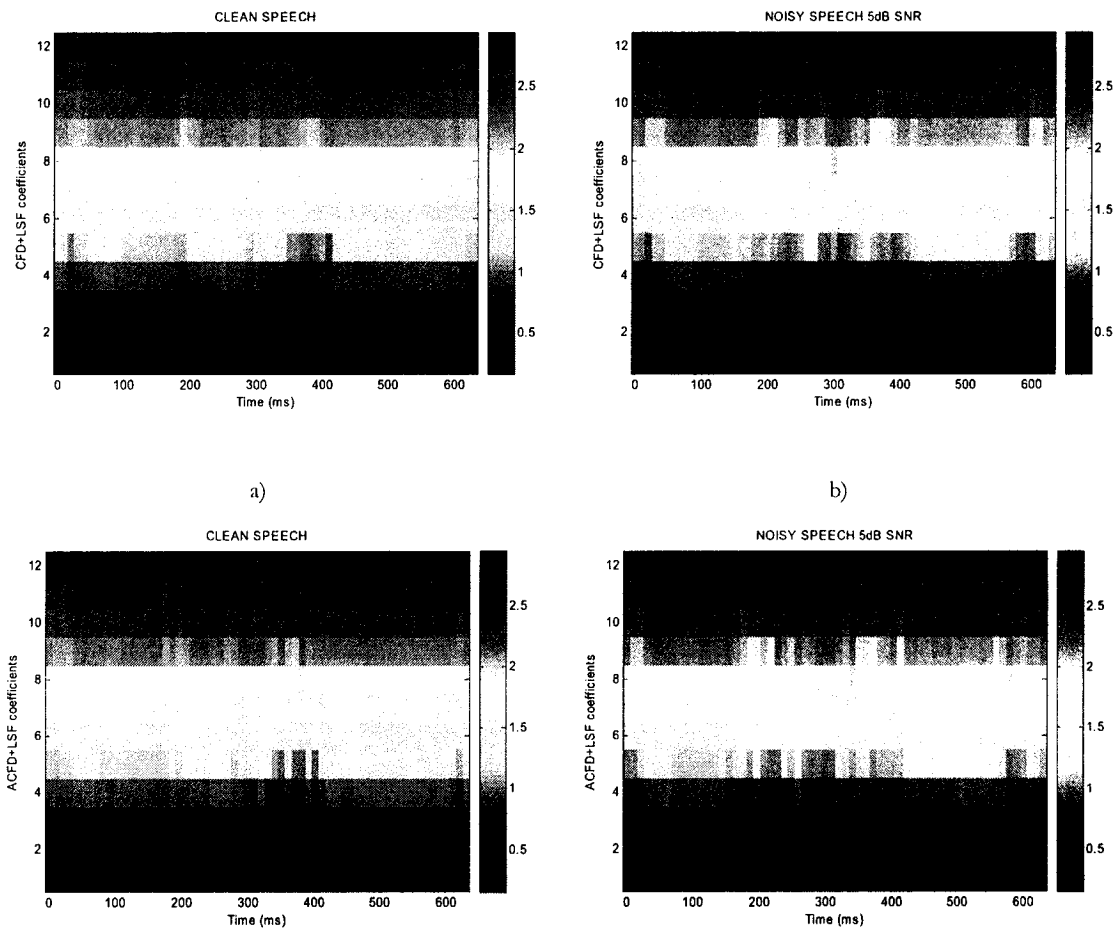


Figure 31 12 LSF coefficients from a), b) 160-CFD cascade spectrum; c), d) 160-ACFD cascade spectrum for the clean and noisy spoken sequence "dark suit"

4.4 Computation Complexity

The computational complexity of the CFD when the number of samples in the frame N is significantly larger than the number of coefficients K is of the order $\sim O(N^2)$, which is more complex than LPC under the same conditions. On the other hand, the ACFD has a complexity of order $\sim O(N)$. This is less complex than LPC, because all that is required is the autocorrelation of the speech frame without the Levinson-Durbin calculation. When $K \approx N$, as in the case of CFD+LPC and CFD+LSF, the CFD complexity grows to $\sim O(N^3)$. With additional calculations needed to derive the cascade spectrum, the IFFT, LPC coefficients (and LSF if required), the method can become computationally expensive. The ACFD+LPC complexity for $K \approx N$ is $\sim O(N^2)$, which is more manageable.

4.5 Summary

In this chapter, the Comb Filter Decomposition (CFD), a new method for robust feature extraction based on harmonic decomposition of the speech signal was presented. An interpretation of normalized autocorrelation in terms of CFD, the ACFD, was given and three feature extraction methods based on the CFD and ACFD were proposed. The chapter concluded with the analysis of the CFD's and ACFD's computational complexity. The next chapter presents simulation results that evaluate the robustness of the new proposed feature extraction methods and their performance as compared to the existing methods.

Chapter 5

SIMULATIONS AND TEST RESULTS

5.1 Setup

Simulations were done in Matlab 7.0. The ASR system was built using HMMs with GMM observation probability functions for the classifier. The system was built using standard Matlab functions in addition to:

- VOICEBOX Toolbox for MFCC feature extraction [48]
- BNT Toolbox for HMM with GMM training [49]
- PLP-RASTA Toolbox for some of the preprocessing [50]

5.1.1 Preprocessing

All the data used for the simulation was taken from the TIMIT speech database using single word utterances without silence at the beginning or end – it was assumed that the data was preprocessed by some Voice Activity Detection (VAD) method. The speech data was downsampled to 8kHz, divided into 20ms frames with 50% overlap. A Hamming window was applied to the frames for all the methods with the exception of the CFD.

For testing the ASR with the input speech at different SNRs, random noise v(n) was added to the speech data. The noise was modeled as a Gaussian process with the following characteristics:

$$v(n) \sim N(0, \sigma_v^2) \tag{5.1}$$

, where the noise variance was calculated from the desired SNR

$$\sigma_v^2 = \frac{\sigma_s^2}{10^{\frac{SNR}{10}}} \tag{5.2}$$

The σ_s^2 is the variance of the clean speech signal, and the SNR is expressed in dBs describing the ratio of clean signal to noise according to the relation:

$$SNR = 10\log_{10}(\frac{\sigma_s^2}{\sigma_v^2}) \tag{5.3}$$

5.1.2 Feature Extraction

The feature extraction and preprocessing methods used in the simulations are identified by the labels that show the order of the processing delimited by '+' for each corresponding method:

- LPC – 12 coefficient LPC
- PLP+LPC – PLP processing followed by 12 coefficient LPC
- RASTA+LPC – RASTA filtering followed by 12 coefficient LPC
- PLP+RASTA+LPC – PLP processing with RASTA filtering followed by 12 coefficient LPC
- MFCC – 30 filter Mel-bank filtering followed by 7 coefficient CC
- MFCC+CMS+CN – 30 filter Mel-bank filtering followed by 7 coefficient CC with CMS and CN
- MFCC+CGN - 30 filter Mel-bank filtering followed by 7 coefficient CC with CGN
- LSF – LSF from 12 coefficient LPC
- SENT – 5 coefficient spectral entropy feature extraction
- CFD – 12 coefficient CFD
- ACFD – 12 coefficient ACFD
- CFD+LPC – 12 coefficient LPC from 160 CFD cascade spectrum
- ACFD+LPC – 12 coefficient LPC from 160 ACFD cascade spectrum
- CFD+LSF –12 coefficient LPC from 160 CFD cascade spectrum followed by LSF
- ACFD+LSF –12 coefficient LPC from 160 ACFD cascade spectrum followed by LSF

Simulations with delta features are marked by D at the end of a feature label (for example 'PLP+LPC+D'). Delta processing is done on the final features with 7-frame windows and the resulting delta vector is appended to the original feature vector resulting in new double length vector.

Simulations with a preemphasis filter are marked by pf at the beginning of a feature label (for example 'pf+PLP+LPC'). The speech data is filtered with a preemphasis filter with coefficient a=0.95 prior to any other processing.

5.1.3 Classification

Recognition was performed on a 10-word dictionary consisting of the words: suit, like, greasy, oily, dark, year, rag, ask, wash, don't. 10 HMMs were created, one for each word class, and the states in the HMMs corresponded to phonemes of the whole phoneme alphabet for the 10-word dictionary. The training of each HMM was done by running the EM algorithm using 20 utterances of the HMM's corresponding word class, where the phoneme label for each frame of speech was available. Testing was done with 100 different utterances of each word by running the Viterbi algorithm on each HMM to find the best state sequence for features from a given test utterance. The word class of the test utterance was declared to be the one corresponding to the HMM that returned highest probability of its best state. The testing was done for input data at various SNRs generated according to the method described in Section 5.1.1.

Figure 32 shows a flowchart of all the possible methods that constituted the ASR system for the simulations. The signal s(n) represents the speech input and the label C_i represents the class of the input sequence as declared by the ASR for the i=1,2,...,10 word vocabulary. The greyed-out boxes in the figure represent the methods that were mentioned in this paper, but not implemented in the simulations.

Since the purpose of the simulations was to evaluate various feature extraction and preprocessing methods in noisy conditions, the results stress the changes of recognition error rates with decreasing SNR, rather than the actual recognition rates for clean speech. There was some effort spent on getting reasonable recognition rates for high SNR input, but the

ASR systems were not optimized to match the performance of current state of the art ASR systems on clean speech. The performance is shown in terms of recognition error rates, which is the percentage of the test words that were misclassified. Increased error rate translates into degrading ASR system performance.

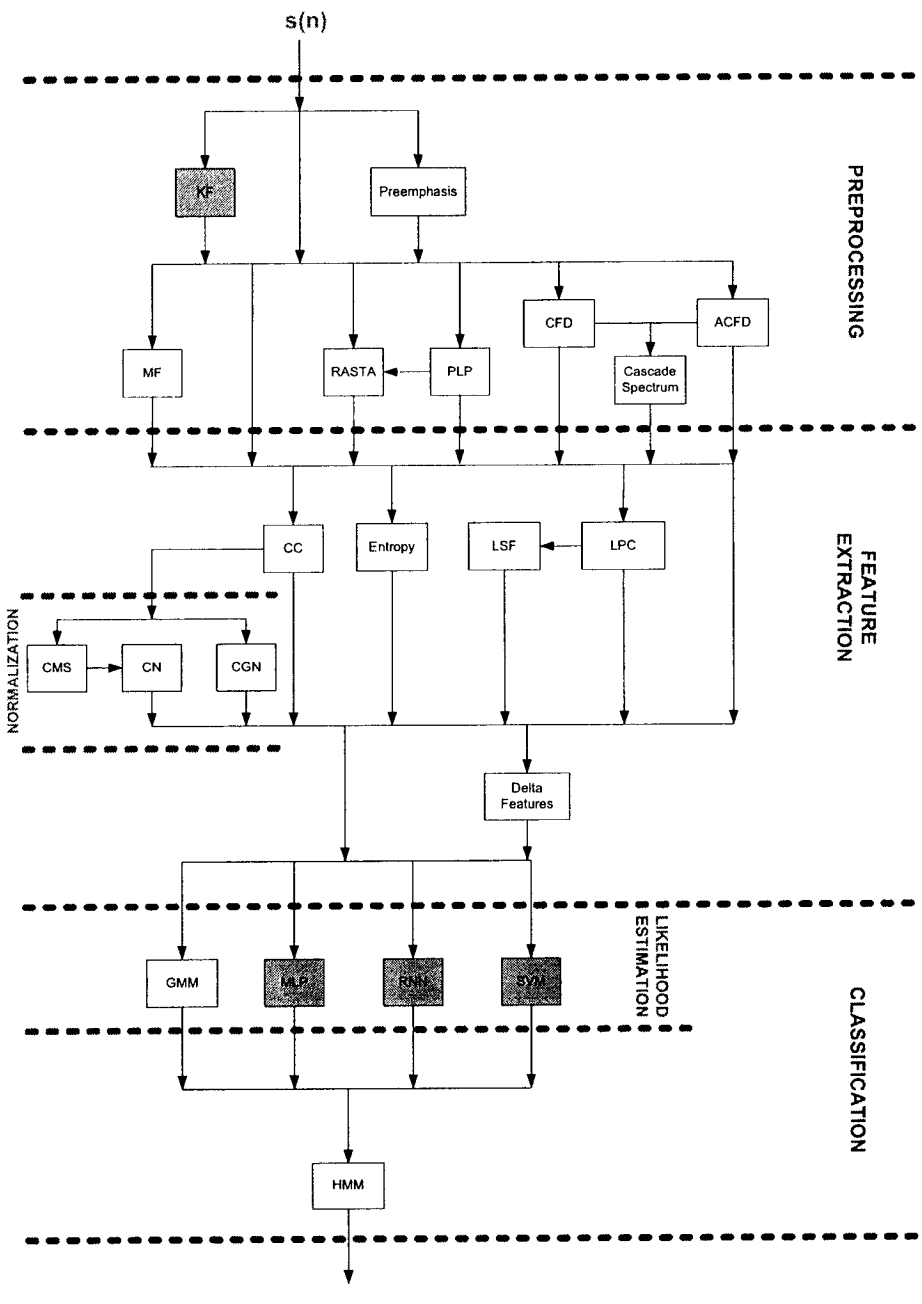


Figure 32 ASR System Flowchart

5.2 Recognition performance

The results are displayed in groups of four simulations where the evaluated ASR included: standard features, preemphasis filtering, delta features, preemphasis with delta features. This was done to facilitate the change of the feature extraction performance for various ASR systems.

5.2.1 LPC vs. MFCC

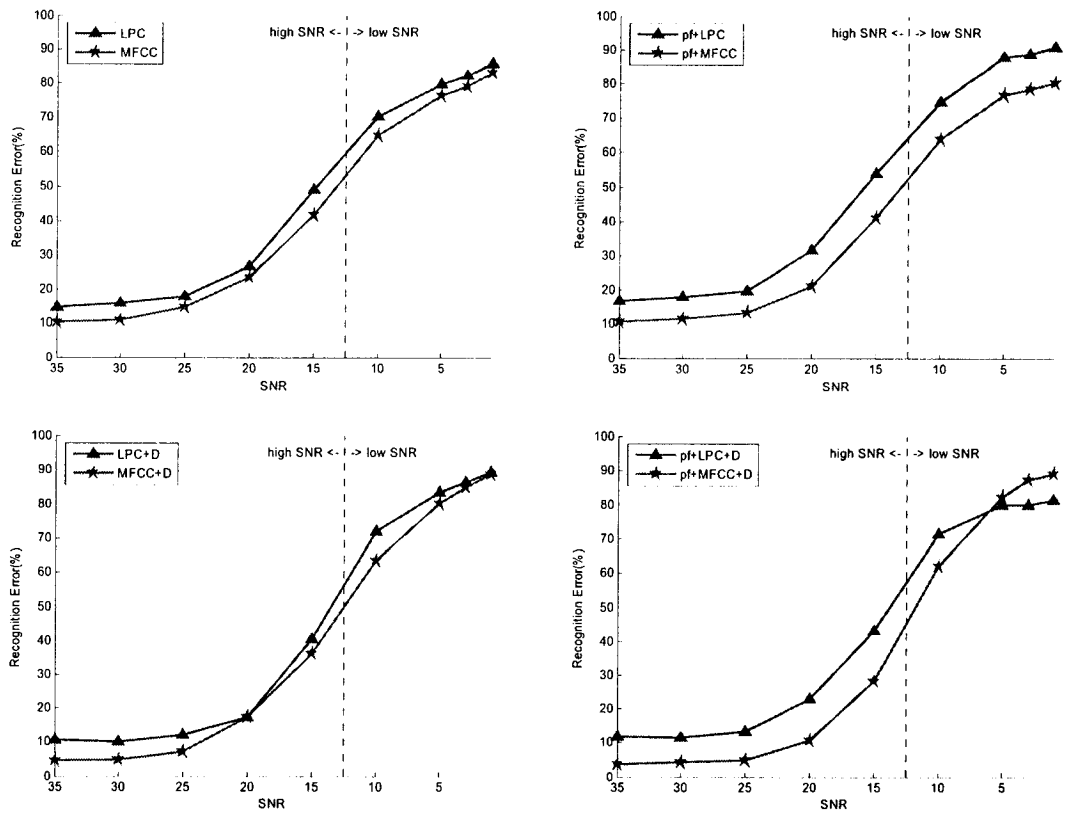


Figure 33 Simulation results for MFCC and LPC

Figure 33 compares the standard LPC and MFCC feature extraction methods. Generally the MFCC gives a better performance (ie. lower error recognition rates). Preemphasis does improve MFCC slightly for high SNR at a cost of slightly higher error rates for low SNR. Delta features improve the ASR performance significantly for clean or high SNR.

5.2.2 LPC with PLP and RASTA

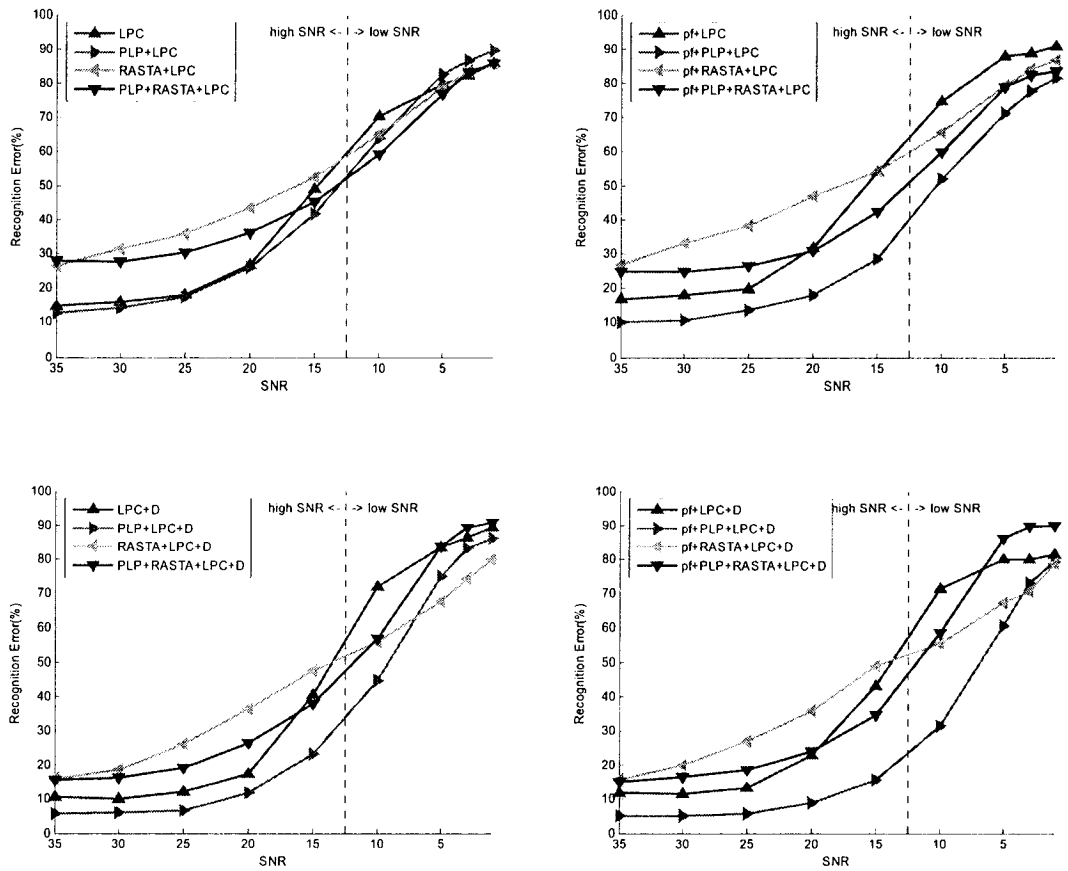


Figure 34 Simulation results for LPC, PLP+LPC, RASTA+LPC, and PLP+RASTA+LPC

Figure 34 shows the performance of robust feature extraction methods based on LPC. PLP preprocessing improves the overall performance of the system, while RASTA filtering improves the recognition rates for low SNR (especially in systems with delta features) at a cost of a significant decrease in performance for high SNR.

5.2.3 LPC vs. LSF

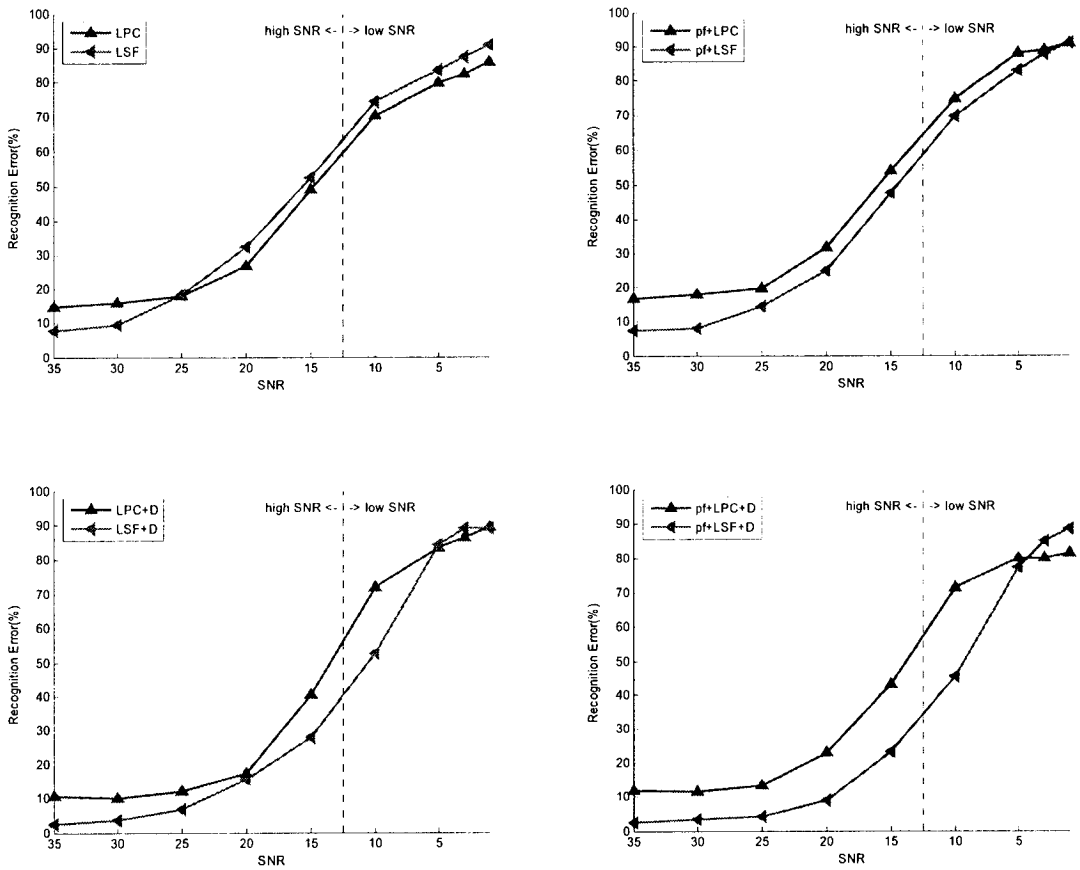


Figure 35 Simulation results for LPC and LSF

Figure 35 compares the performance of LPC to LSF. Generally LSF improves performance of the ASR system, especially in the high SNR.

5.2.4 MFCC with Cepstral normalization

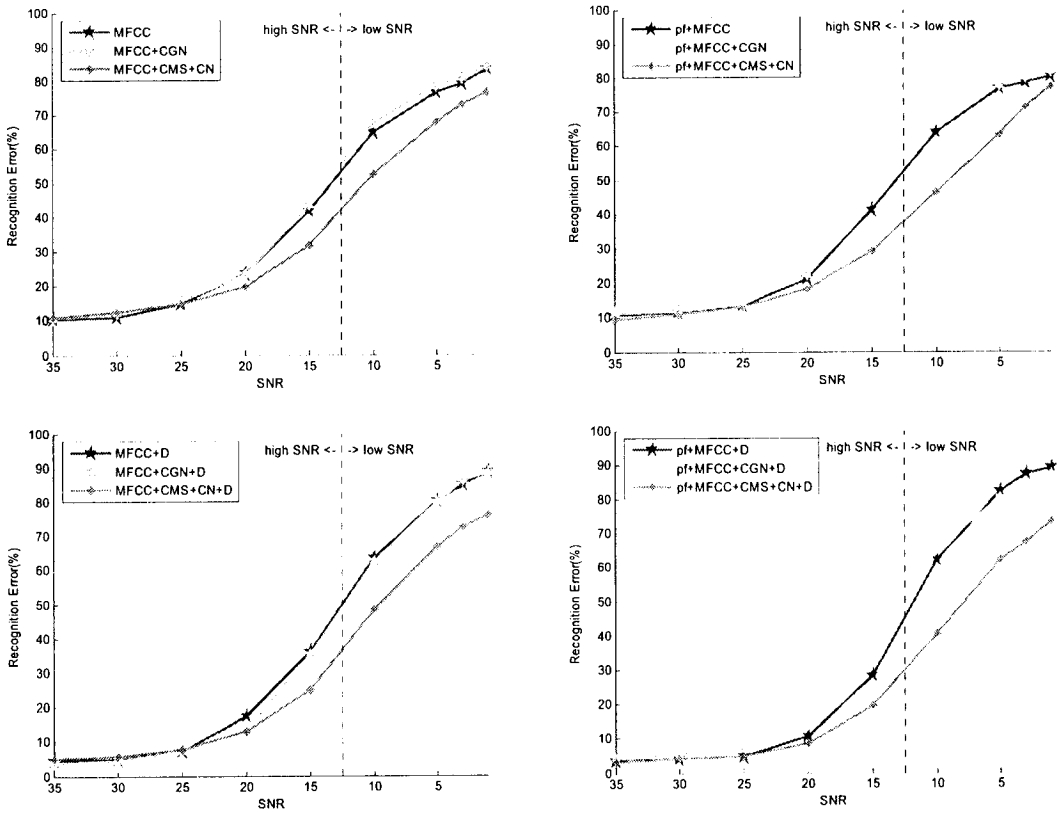


Figure 36 Simulation results for MFCC, MFCC+CMS+CN, and MFCC+CGN

Figure 36 shows the performance of robust feature extraction methods based on cepstral normalizations. While MFCC+CGN does not seem to give any improvement over the standard MFCC, MFCC+CMS+CN improves the system performance significantly for low SNR, while maintaining the same performance as the standard MFCC in high SNR.

5.2.5 Spectral Entropy vs. LPC and MFCC

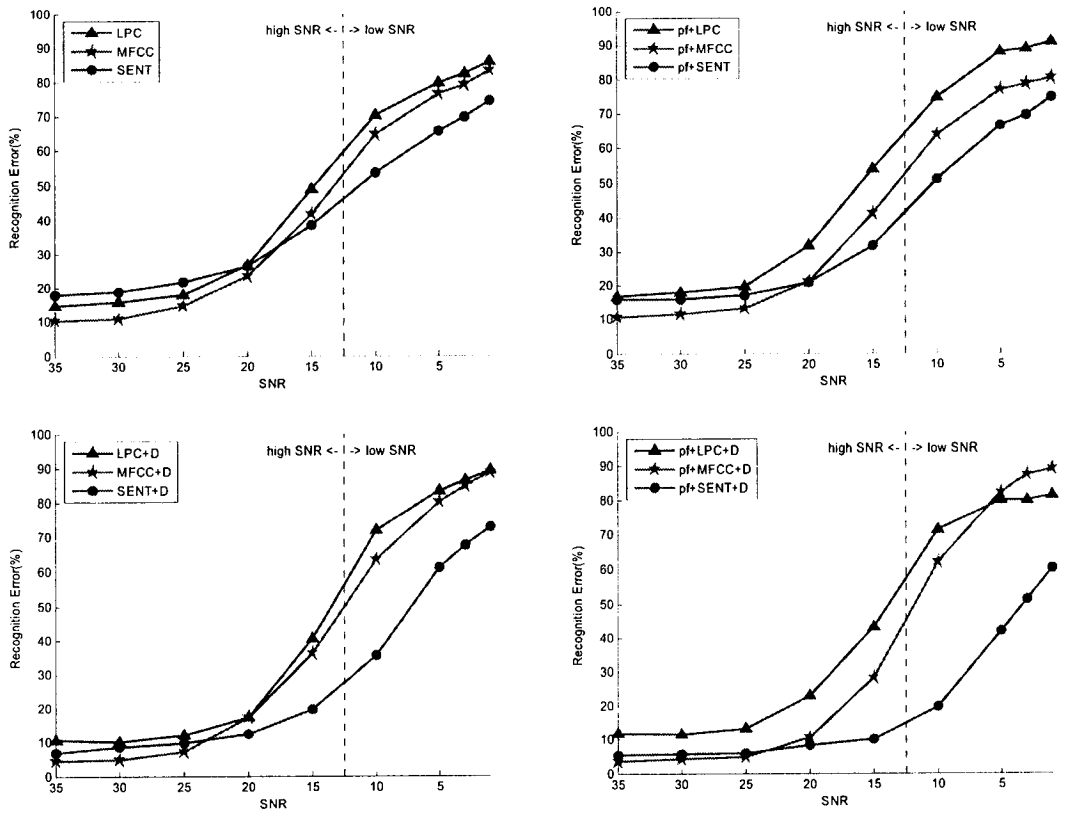


Figure 37 Simulation results for MFCC, LPC, and SENT

Figure 37 compares the spectral entropy (SENT) robust feature extraction to the standard LPC and MFCC methods. The SENT generally matches the LPC in high SNR and it outperforms both LPC and MFCC for low SNR. In the ASR system with preemphasis and delta features, the SENT matches the MFCC in high SNR and gives a very significant improvement in low SNR.

5.2.6 CFD vs. LPC and MFCC

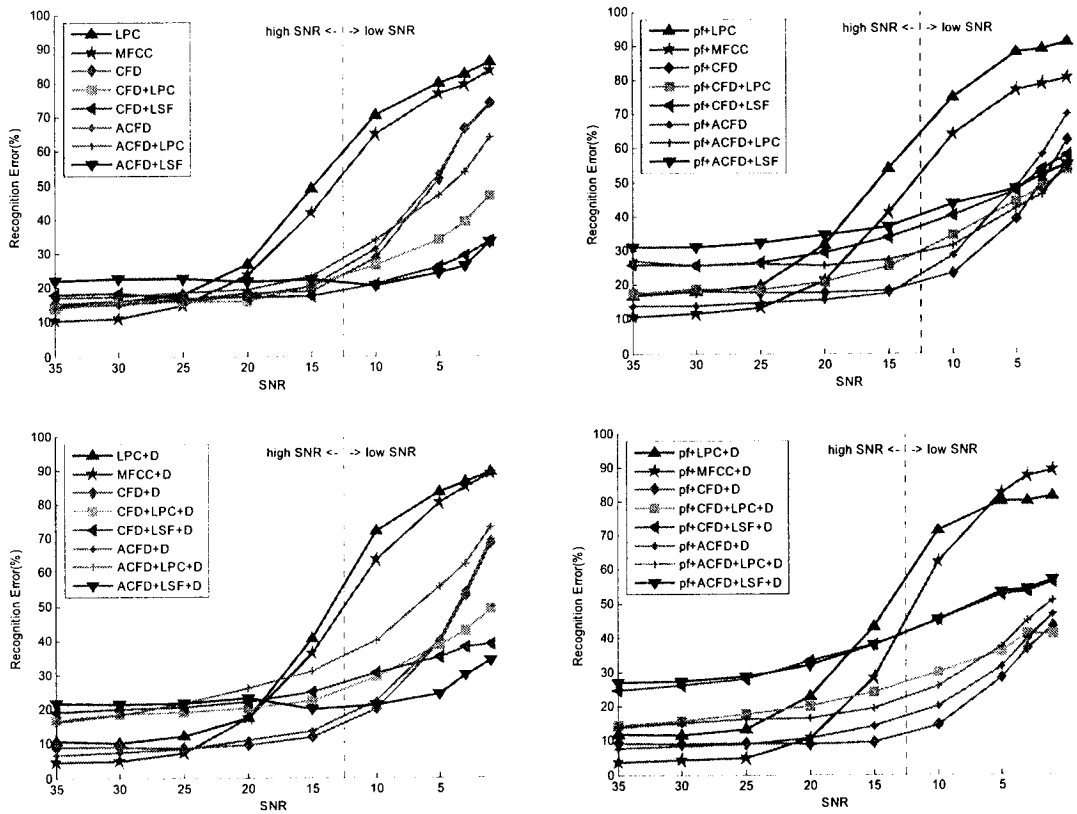


Figure 38 Simulation results for LPC, MFCC, CFD, ACFD, CFD+LPC, ACFD+LPC, CFD+LSF and ACFD+LSF

Figure 38 shows the performance of the proposed CFD and ACFD methods as compared to the standard LPC and MFCC. Three CFD-based feature extraction methods are shown – CFD, CFD+LPC and CFD+LSF. The same is done for ACFD-based features. In the ASR system without preemphasis and/or delta features, the CFD+LSF significantly improves the performance of the system in low SNR while matching LPC in clean conditions. ACFD+LSF slightly outperforms CFD+LSF in noisy conditions, but it has a higher error rate in high SNR. Performance of both CFD+LSF and ACFD+LSF declines for ASR systems with preemphasis and/or delta features. In those systems, the CFD and ACFD features give the best overall performance.

5.2.7 Most robust features

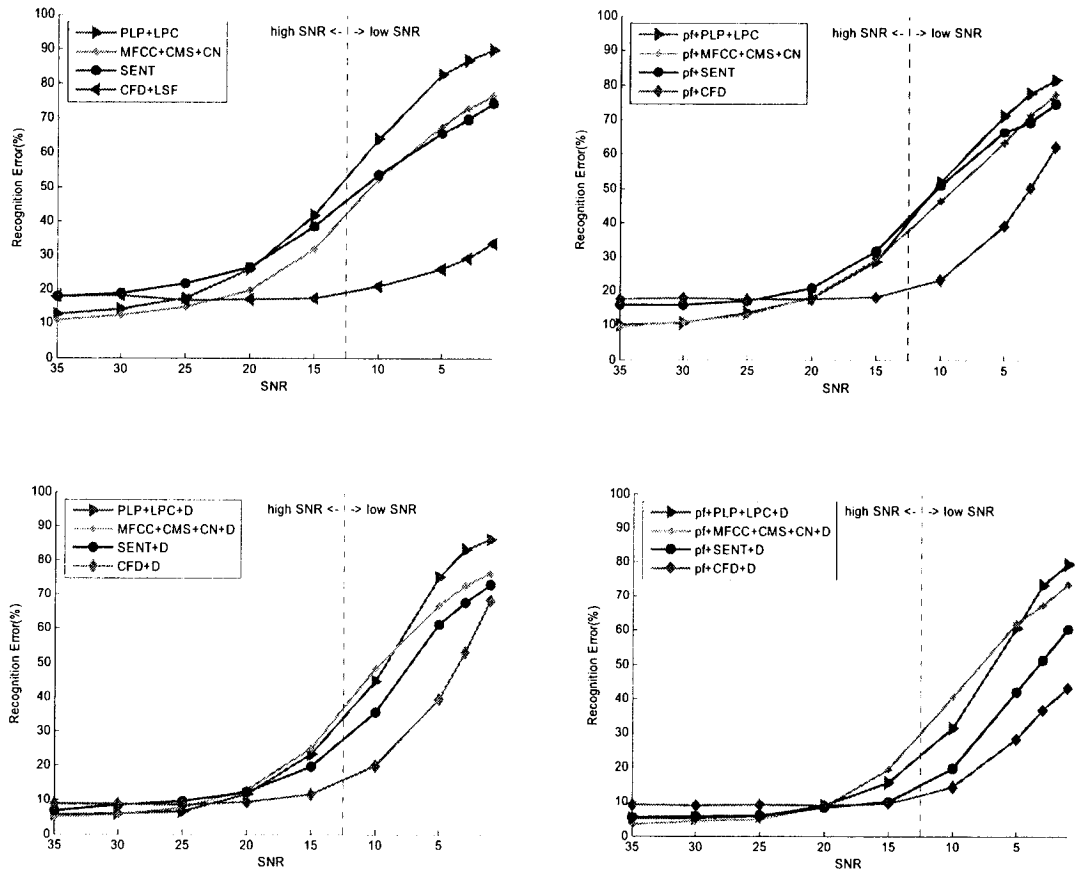


Figure 39 Simulation results for PLP+LPC, MFCC+CMS+CN, SENT, CFD+LSF and CFD

Figure 39 compares the best robust feature extraction methods from the previous figures. CFD-based features give the best performance in low SNR, while giving around 8% higher error rates than MFCC+CMS+CN for high SNR. CFD+LSF is most robust for the ASR without preemphasis and/or delta features, while CFD is most robust in other scenarios.

Table 1 shows the recognition error rates for all the methods implemented in the simulations.

Table 1. Percent error recognition rates for 8kHz data

Feature type	SNR (dB)								
	35	30	25	20	15	10	5	3	1
LPC	14.9	15.9	17.9	26.8	49.1	70.4	79.6	82.4	85.8
pf+LPC	16.8	18.1	19.6	31.7	54.0	74.8	87.8	88.8	90.7
LPC+D	10.7	10.1	12.2	17.5	40.6	72.1	83.4	86.5	89.3
pf+LPC+D	11.9	11.5	13.2	22.9	43.1	71.5	80.0	79.9	81.4
PLP+LPC	12.7	14.1	17.4	25.9	41.7	63.8	82.6	86.7	89.6
pf+PLP+LPC	10.1	10.8	13.5	18.0	28.6	51.9	71.3	77.5	81.5
PLP+LPC+D	5.6	5.9	6.7	11.7	23.1	44.6	75.0	83.1	86.1
pf+PLP+LPC+D	5.0	5.0	5.6	8.9	15.6	31.4	60.8	73.1	79.3
RASTA+LPC	26.5	31.4	35.8	43.4	52.4	65.0	78.7	83.5	85.5
pf+RASTA+LPC	26.8	33.2	38.1	46.9	54.1	65.6	79.5	84.0	86.6
RASTA+LPC+D	15.8	18.6	26.3	36.4	47.4	55.9	67.7	74.3	80.0
pf+RASTA+LPC+D	15.6	19.9	27.1	35.7	49.0	55.4	67.3	70.6	78.7
PLP+RASTA+LPC	27.9	27.7	30.4	36.2	45.1	59.1	76.8	83.3	85.8
pf+PLP+RASTA+LPC	24.7	24.7	26.4	30.9	42.3	59.7	78.8	82.4	83.4
PLP+RASTA+LPC+D	15.5	16.2	19.1	26.6	38.0	56.9	83.7	89.2	90.8
pf+PLP+RASTA+LPC+D	15.0	16.5	18.6	24.1	34.7	58.6	86.0	89.5	90.0
LSF	7.7	9.5	18.4	32.2	52.5	74.3	83.5	87.3	90.9
pf+LSF	7.5	7.9	14.4	24.9	47.6	69.7	82.8	87.5	90.8
LSF+D	**2.4**	3.6	6.9	15.5	27.9	52.8	84.5	88.9	89.1
pf+LSF+D	2.5	**3.4**	**4.1**	8.8	23.3	45.6	77.4	84.9	88.5
MFCC	10.5	11.1	14.9	23.5	41.8	64.8	76.6	79.0	83.2
pf+MFCC	10.8	11.6	13.4	21.1	41.2	63.8	76.9	78.6	80.2
MFCC+D	4.6	4.8	7.1	17.4	36.2	63.7	80.2	84.8	88.6
pf+MFCC+D	3.6	4.2	4.8	10.6	28.3	62.2	82.4	87.2	89.1
MFCC+CMS+CN	11.1	12.5	14.7	19.8	31.7	52.4	67.8	73.0	76.4
pf+MFCC+CMS+CN	9.4	11.0	12.9	18.2	29.0	46.2	63.4	71.6	77.2
MFCC+CMS+CN+D	5.1	5.7	7.7	13.1	25.0	48.3	66.9	72.8	76.1
pf+MFCC+CMS+CN+D	3.5	4.2	4.9	8.7	19.5	40.4	62.2	67.5	73.6
MFCC+CGN	10.6	12.2	15.7	23.1	43.3	67.6	78.3	80.5	84.0
pf+MFCC+CGN	11.4	11.7	13.8	22.4	46.8	67.8	77.5	80.1	81.7
MFCC+CGN+D	4.0	4.5	7.2	13.4	35.2	62.3	79.9	86.0	88.5
pf+MFCC+CGN+D	4.2	5.3	7.2	14.2	33.1	65.6	80.4	84.7	86.9
SENT	18.1	18.8	21.9	26.6	38.4	53.7	65.5	69.7	74.3
pf+SENT	16.0	15.9	17.1	21.0	31.7	51.0	66.6	69.5	74.8
SENT+D	7.0	8.6	9.7	12.4	19.8	35.6	61.4	67.8	72.9
pf+SENT+D	5.3	5.6	6.0	**8.3**	10.2	19.6	41.9	51.2	60.5
CFD	15.3	16.1	16.4	18.2	19.0	28.4	51.5	66.1	73.8
pf+CFD	17.6	18.1	17.8	17.8	18.3	23.3	39.1	50.2	62.2
CFD+D	8.9	8.8	8.7	9.5	11.8	19.9	39.4	53.0	68.3
pf+CFD+D	9.1	9.0	9.3	8.8	**9.5**	**14.6**	28.2	36.7	43.0
ACFD	14.6	15.1	16.1	17.3	20.2	31.0	53.2	65.8	73.1
pf+ACFD	13.8	14.0	14.9	15.5	17.3	28.5	48.5	57.9	69.7
ACFD+D	6.6	7.5	8.4	11.0	13.6	22.3	40.3	54.3	69.4

pf+ACFD+D	7.7	8.2	9.0	10.6	14.1	20.0	31.5	39.6	46.6
CFD+LPC	13.9	15.9	15.9	15.9	20.6	26.5	33.9	39.0	46.6
pf+CFD+LPC	17.5	18.9	18.6	20.8	25.4	34.3	44.2	48.5	53.4
CFD+LPC+D	16.9	18.5	19.1	20.4	22.3	29.5	38.5	42.8	48.9
pf+CFD+LPC+D	14.4	15.6	17.6	20.1	24.1	30.0	35.7	41.0	40.7
ACFD+LPC	17.2	17.3	18.6	19.5	23.1	33.7	46.9	53.8	63.8
pf+ACFD+LPC	27.1	25.5	26.1	25.5	27.4	31.3	42.2	46.4	55.5
ACFD+LPC+D	16.1	18.3	21.8	26.1	31.2	40.0	55.8	62.4	73.3
pf+ACFD+LPC+D	13.9	15.1	16.2	16.5	19.4	25.9	37.4	45.0	50.7
CFD+LSF	18.0	18.4	16.9	17.1	17.5	20.8	25.8	29.0	33.6
pf+CFD+LSF	25.8	25.6	26.6	29.5	33.7	40.1	47.6	53.6	57.8
CFD+LSF+D	19.1	20.1	20.6	22.2	24.9	30.6	35.0	37.8	38.7
pf+CFD+LSF+D	24.7	26.2	27.9	33.3	38.0	45.2	52.4	53.3	56.2
ACFD+LSF	22.0	22.6	22.6	21.9	22.3	20.6	**24.0**	**26.1**	**33.2**
pf+ACFD+LSF	31.1	31.2	32.2	34.6	37.1	43.7	48.2	51.9	55.1
ACFD+LSF+D	21.9	21.5	21.7	23.3	20.0	21.2	24.4	30.1	34.1
pf+ACFD+LSF+D	27.1	27.4	28.8	32.1	37.8	45.2	53.3	54.2	56.8

Table 1 shows that pf+LSF+D gives the lowest error rates for clean speech and ACFD+LSF gives the lowest error rates for noisy speech. In between, the best performance is obtained with pf+SENT+D and CFD+D features. Generally, preemphasis is beneficial only in high SNR and delta features improve recognition rates in both high and low SNR (with the exception of ACFD+LSF, which performs much better without delta features).

5.3 Execution time performance

Figure 40 shows the results of the execution time benchmark for all the feature extraction methods used in the simulations with the setup/parameters as described in Section 5.1. The execution time is normalized by the time taken to perform the standard LPC analysis under the same conditions. The benchmark was done in MATLAB 7.0 on the same data that was used for the recognition performance evaluation in Section 5.2.

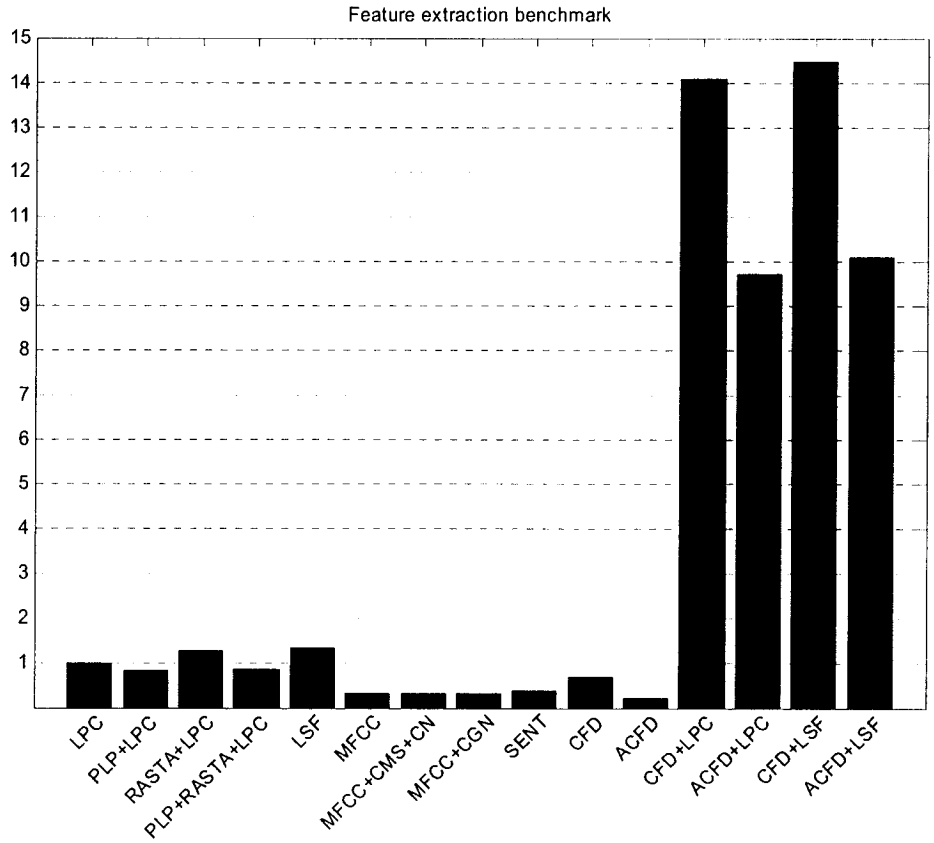


Figure 40 Execution time for different feature extraction methods as a fraction of time taken for standard LPC analysis.

Execution time benchmarking confirms that ACFD-based features are less computationally intensive than CFD-based ones. The 12-coefficient CFD and ACFD feature extraction is faster than the standard LPC, and in the case of ACFD, it matches the execution time of the 30 Mel-bank 7-coefficient MFCC. On the other hand the CFD+LPC and CFD+LSF take almost 15 times longer than the standard LPC, while the execution time for ACFD+LPC and ACFD+LSF is about 10 times that of the standard LPC. The long execution time is the result of the computationally intensive 160-coefficient CFD and ACFD analysis, and formation of the cascade spectra followed by the LPC or LSF analysis.

5.4 Summary

This chapter presented the simulation results for the proposed CFD and ACFD-based feature extraction methods with comparison to selected existing standard and robust features. The new features show resilience to additive white Gaussian noise and improved overall performance of the ASR system in various SNRs. Results for the execution time benchmarking were also shown.

Chapter 6

CONCLUSION

Comb Filtering Decomposition was presented as a viable method for speech processing in ASR systems. Two flavors of the analysis were introduced, the CFD and ACFD, with three feature extraction methods from the CFD and ACFD coefficients. For a small vocabulary system and an environment where the noise is additive and white, the proposed CFD+LSF and ACFD+LSF feature extraction gives over 50% recognition improvement in conditions of very low SNR as compared to the standard MFCC and LPC. The CFD and ACFD features generally match, and sometimes outperform, the Spectral Entropy features, which were reported to be the best existing features for coping with noise. For clean speech, on average, the CFD+LSF and ACFD+LSF increase the recognition error by about 10% (5% for the CFD and ACFE) as compared to the MFCC. The CFD+LPC and ACFD+LPC features show a similar, although slightly weaker, performance than the CFD+LSF and ACFD+LSF, respectively.

Whereas the computational complexity of the CFD and ACFD features is smaller than that of the standard LPC, the CFD+LPC, CFD+LSF, ACFD+LPC and ACFD+LSF are computationally intensive to the point where the execution time might be impractical for some of the current ASR applications. Future work should address the issue of the complexity and/or possibly suggest other feature extraction methods from the CFD/ACFD coefficients. For instance, a method that would place a larger emphasis on the low frequencies of the speech spectrum might improve recognition rates. Another future improvement to the CFD analysis could be shaping the frequency response of the comb-filters to obtain a better fit to the envelope of the speech spectrum rather than using the comb filters with a constant peak magnitude. This could result in better recognition accuracy in both clean and noisy conditions.

Appendix A

AUTOCORRELATION METHOD FOR CALCULATION OF LPC COEFFICIENTS

The LPC analysis models the n^{th} sample of frame speech t, $s_t(n)$, as a linear combination of previous p samples $s_t(n-1),...,s_t(n-p)$ multiplied by p LPC coefficients x_{tk} .

$$s_t(n) = \sum_{k=1}^{p} x_{tk} s_t(n-k) + G_{t_n} u_t(n) \tag{A.1}$$

Multiplying both sides of the equation (A.1) in turn by $s_t(n-1),...,s_t(n-p)$ results in p equations

$$s_t(n)s_t(n-1) = \sum_{k=1}^{p} x_{tk} s_t(n-k)s_t(n-1) + G_{t_n} u_t(n)s_t(n-1) \tag{A.2}$$

$$s_t(n)s_t(n-2) = \sum_{k=1}^{p} x_{tk} s_t(n-k)s_t(n-2) + G_{t_n} u_t(n)s_t(n-2)$$

$$\vdots$$

$$s_t(n)s_t(n-p) = \sum_{k=1}^{p} x_{tk} s_t(n-k)s_t(n-p) + G_{t_n} u_t(n)s_t(n-p)$$

Taking the expectation of both sides of all the equations, and assuming that the excitation signal at time n is uncorrelated with past speech samples, we get the relation:

$$
\begin{aligned}
r_t(-1) &= \sum_{k=1}^{p} x_{t_k} r_t(k-1) \\
r_t(-2) &= \sum_{k=1}^{p} x_{t_k} r_t(k-2) \\
&\cdot \\
&\cdot \\
&\cdot \\
r_t(-3) &= \sum_{k=1}^{p} x_{t_k} r_t(k-p)
\end{aligned}
\tag{A.3}
$$

where

$$
r_t(m) = \sum_{n=0}^{N-1} s_t(n) s_t(n+m) \tag{A.4}
$$

is the estimation of the autocorrelation of the speech signal. Expressing the set of equations (A.3) in matrix form and exploiting the fact that the autocorrelation is an even function ($r_t(m) = r_t(-m)$), we get:

$$
\mathbf{R}_t \mathbf{x}_t = \mathbf{r}_t \tag{A.5}
$$

$$
\mathbf{R}_t = \begin{bmatrix}
r_t(0) & r_t(1) & r_t(2) & \dots & r_t(p-1) \\
r_t(1) & r_t(0) & r_t(1) & \dots & r_t(p-2) \\
\cdot & \cdot & \cdot & \dots & \cdot \\
\cdot & \cdot & \cdot & \dots & \cdot \\
\cdot & \cdot & \cdot & \dots & \cdot \\
r_t(p-1) & r_t(p-2) & r_t(p-3) & \dots & r_t(0)
\end{bmatrix}
$$

$$
\mathbf{x}_t = \begin{bmatrix} x_{t_1} & x_{t_2} & \dots & x_{t_p} \end{bmatrix}^T
$$

$$
\mathbf{r}_t = \begin{bmatrix} r_t(1) & r_t(2) & \dots & r_t(p) \end{bmatrix}^T
$$

Multiplying both sides by the inverse of $\mathbf{R}_t$ allows to solve for the LPC coefficient vector $\mathbf{x}_t$.

Appendix B

HMM TRAINING AND CLASSIFICATION

EM Algorithm for HMM Parameter Estimation with GMM Observation Model

The Expectation Maximization (EM) algorithm is a maximum likelihood (ML) method of estimating model parameters that gives the highest probability of observing the training data. An HMM model is created for each class of word to be recognized by an ASR system. It is assumed that a set of E training samples W ϵ $\{w_1, w_2, ..., w_E\}$ with the pronunciation of the word corresponding to the HMM is available and that the correspondence of feature vector to phoneme labels (or HMM state) constituting a word sequence is known: $w_1=\{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_L\}\sim\{q_1,q_2,...,q_L\}$. L, the number of feature vectors, may vary from pronunciation to pronunciation. An HMM consists of M states corresponding to a M phoneme alphabet $q_t\epsilon\{\omega_1,\omega_2,...,\omega_M\}$. The training attempts to find parameters HMM $\lambda=(\mathbf{A},\mathbf{B},\pi)$ that maximize the probability of the desired state sequence $Q=\{q_1,q_2,...,q_L\}$ when the observed sequence is $X=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_L\}$. A reasonable estimate of the initial value of state transition matrix **A**, observation probability matrix **B** and initial probability matrix π is required, and it can be derived from the known feature vector to phoneme relation (for details on the definitions of **A**, π see equations (2.22) and (2.24); for details on the definition of **B** see equations (2.23) and (2.29)).

The following quantities are defined to simplify the notation [11],[12],[13],[14]. The probability of seeing the partial sequence $\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_t$ and ending up in state ω_i at time t is:

$$\alpha_i(t) = p(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_t,q_t=\omega_i|\lambda) \tag{B.1}$$
$$\boldsymbol{\alpha}_t=\left[\alpha_1(t)\quad\alpha_2(t)\quad\ldots\quad\alpha_M(t)\right]^T \qquad \mathrm{M}\times 1$$

The vector $\boldsymbol{\alpha}_t$ can be calculated as follows (the symbol ' • ' represents an element by element multiplication):

$$\begin{aligned} \boldsymbol{\alpha}_1 &= \boldsymbol{\pi} \cdot \mathbf{b}_{\mathbf{x}_1} \\ \boldsymbol{\alpha}_{t+1} &= \mathbf{A}^T \boldsymbol{\alpha}_t \cdot \mathbf{b}_{\mathbf{x}_{t+1}} \end{aligned} \tag{B.2}$$

The probability of ending with partial sequence $\mathbf{x}_{t+1},\mathbf{x}_{t+2},....,\mathbf{x}_L$ given that the start point was in state ω_i at time t is:

$$\begin{aligned} \beta_i(t) &= p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots, \mathbf{x}_T | q_t = \omega_i, \lambda) \\ \boldsymbol{\beta}_t &= \left[\beta_1(t) \quad \beta_2(t) \quad \ldots \quad \beta_M(t)\right]^T \quad \mathrm{M} \times 1 \end{aligned} \tag{B.3}$$

The value of β_t can be calculated recursively:

$$\begin{aligned} \boldsymbol{\beta}_L &= \mathbf{1} \\ \boldsymbol{\beta}_t &= \mathbf{A}(\boldsymbol{\beta}_{t+1} \cdot \mathbf{b}_{\mathbf{x}_{t+1}}) \end{aligned} \tag{B.4}$$

The probability of the observation sequence $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\}$ for a given HMM model can be calculated independently from α and β:

$$\begin{aligned} p(X \mid \lambda) &= \textstyle\sum_{i=1}^{M} \alpha_i(L) \\ p(X \mid \lambda) &= \textstyle\sum_{i=1}^{M} \beta_i(1) \pi_i b_i(\mathbf{x}_1) \end{aligned} \tag{B.5}$$

The probability of being in state ω_i at time t for the sequence $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\}$ is:

$$\begin{aligned} \gamma_i(t) &= p(q_t = \omega_i \mid X, \lambda) \\ \boldsymbol{\gamma}_t &= \left[\gamma_i(t) \quad \gamma_i(t) \quad \ldots \quad \gamma_i(t)\right]^T \qquad \mathrm{M} \times 1 \end{aligned} \tag{B.6}$$

The vector $\boldsymbol{\gamma}_t$ can be calculated from the relation:

$$\boldsymbol{\gamma}_t = \frac{\boldsymbol{\alpha}_t \cdot \boldsymbol{\beta}_t}{\boldsymbol{\alpha}_t^T \boldsymbol{\beta}_t} \tag{B.7}$$

The probability of being in state ω_i at time t and being in state ω_j at time t+1 for the sequence $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\}$ is:

$$\xi_{ij}(t) = p(q_t = \omega_i, q_{t+1} = \omega_j \mid X, \lambda) \tag{B.8}$$

$$\xi_t = \begin{bmatrix} \xi_{11}(t) & \xi_{12}(t) & \cdots & \xi_{1M}(t) \\ \xi_{21}(t) & \xi_{22}(t) & \cdots & \xi_{2M}(t) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \xi_{M1}(t) & \xi_{M2}(t) & \cdots & \xi_{MM}(t) \end{bmatrix} \qquad M \times 1$$

The matrix ζ_t can be calculated from the relation:

$$\xi_{ij}(t) = \frac{\gamma_i(t) a_{ij} b_j(\mathbf{x}_{t+1}) \beta_j(t+1)}{\beta_i(t)} \tag{B.9}$$

Recalling that the observation probability distribution is modeled by a GMM model, the probability that the g^{th} component of the i^{th} mixture produces observation $\mathbf{x}_t$ is defined as:

$$\varphi_{ig}(t) = \gamma_i(t) \frac{c_{ig}\, p(\mathbf{x}_t | \boldsymbol{\mu}_{ig}, \boldsymbol{\Sigma}_{ig})}{b_i(\mathbf{x}_t)} = \gamma_i(t) \frac{c_{ig}\, p(\mathbf{x}_t | \boldsymbol{\mu}_{ig}, \boldsymbol{\Sigma}_{ig})}{\sum_{g=1}^{G} c_{ig}\, p(\mathbf{x}_t | \boldsymbol{\mu}_{ig}, \boldsymbol{\Sigma}_{ig})} \tag{B.10}$$

$$\varphi_t = \begin{bmatrix} \varphi_{11}(t) & \varphi_{12}(t) & \cdots & \varphi_{1G}(t) \\ \varphi_{21}(t) & \varphi_{22}(t) & \cdots & \varphi_{2G}(t) \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \varphi_{M1}(t) & \varphi_{M2}(t) & \cdots & \varphi_{MG}(t) \end{bmatrix}$$

Assuming there are K pronunciations of a given word class available for the training, each pronunciation k, for k=1,2,...K, contains L^k frames, and therefore L^k feature vectors. The progression of probabilities γ_t^k, ζ_t^k, and φ_t^k are calculated for each pronunciation for $t=1,2,...L^k$, and the update equations for the initial and state transition probabilities are given by:

$$\pi^{new} = \frac{1}{K} \sum_{k=1}^{K} \gamma_1^k \tag{B.11}$$

$$a_{ij}^{new} = \frac{\sum_{k=1}^{K}\sum_{t=1}^{L^k-1}\xi_{ij}^k(t)}{\sum_{k=1}^{K}\sum_{t=1}^{L^k-1}\gamma_i^k(t)}$$

$$c_{ig}^{new} = \frac{\sum_{k=1}^{K}\sum_{t=1}^{L^k}\varphi_{ig}^k(t)}{\sum_{k=1}^{K}\sum_{t=1}^{L^k}\gamma_i^k(t)}$$

$$\mu_{ig}^{new} = \frac{\sum_{k=1}^{K}\sum_{t=1}^{L^k}\varphi_{ig}^k(t)\mathbf{x}_t}{\sum_{k=1}^{K}\sum_{t=1}^{L^k}\varphi_{ig}^k(t)}$$

$$\Sigma_{ig}^{new} = \frac{\sum_{k=1}^{K}\sum_{t=1}^{L^k}\varphi_{ig}^k(t)(\mathbf{x}_t-\boldsymbol{\mu}_{ig}^{new})(\mathbf{x}_t-\boldsymbol{\mu}_{ig}^{new})^T}{\sum_{t=1}^{L^k}\varphi_{ig}^k(t)}$$

Viterbi algorithm

The Viterbi algorithm finds the best state sequence $Q=\{q_1,q_2,...,q_L\}$ in the HMM $\lambda=(\mathbf{A},\mathbf{B},\pi)$ for a given sequence of feature vectors $X=\{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_L\}$. It is similar to the forward calculation procedure shown in equations (B.1) and (B.2) [11]. The highest probability of being in state ω_i at time t is defined as:

$$\delta_i(t) = \max[P(q_1,q_2,\ldots q_{t-1},q_t=\omega_i,\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_t|\lambda)] \qquad \text{(B.12)}$$
$$\boldsymbol{\delta}_t = [\delta_1(t) \quad \delta_2(t) \quad \ldots \quad \delta_M(t)]$$

The calculations of $\delta_i(t)$ for t=1,..,L are as follows:

$$\delta_i(1) = \pi_i b_i(\mathbf{x}_1) \qquad \text{(B.13)}$$
$$\delta_j(t) = (\max_{1\le i\le M}[\delta_i(t-1)a_{ij}])b_j(\mathbf{x}_{t+1})$$
$$\text{where } t = 2,3,\ldots,L \qquad i,j = 1,2,...,M$$

Having calculated the final vector $\boldsymbol{\delta}_l$, the maximum probability for a sequence X modeled by the HMM λ is given by:

$$P^* = \max_{1 \le i \le M} [\delta_i(L)] \tag{B.14}$$

To find the sequence of states that gives P*, the argument that maximizes $\delta_j(t)$ in equation (B.13) is saved in variable $\phi_j(t)$:

$$\begin{gathered} \phi_i(1) = 0 \\ \phi_j(t) = \arg_i \max_{1 \le i \le M} [\delta_i(t-1) a_{ij}] \\ \text{where } t = 2,3,\ldots,L \qquad i,j = 1,2,\ldots,M \\ \boldsymbol{\phi}_t = \left[\phi_1(t) \quad \phi_2(t) \quad \ldots \quad \phi_M(t)\right] \end{gathered} \tag{B.15}$$

The state sequence giving the probability P* can be calculated recursively from the relation:

$$\begin{aligned} q_L^* &= \arg_i \max_{1 \le i \le M} [\delta_i(L)] \\ q_t^* &= \phi_i(t+1)(q_{t+1}^*) \end{aligned} \tag{B.16}$$

REFERENCES

[1] Jelinek F., "Continuous speech recognition by statistical methods", *Proceedings of the IEEE*, Vol.64:532:556, 1976.

[2] Gauvain J-L, Lamer L., "Large-Vocabulary Continuous Speech Recognition: Advances and Applications", *Proceedings of the IEEE*, Vol.88, No.8:1181-1200, 2000.

[3] O'Shaughnessy D., "Interacting With Computers by Voice: Automatic Speech Recognition and Synthesis", *Proceedings of the IEEE*, Vol.91, No.9:1272-1305, 2003.

[4] Quatieri T. F., Discrete-Time Speech Signal Processing: Principles and Practice, Prentice Hall, Upper Saddle River, 2002.

[5] O'Shaughnessy D., Speech Communications: human and machine, New York, IEEE Press, 2000.

[6] Rabiner L., Juang B.H., Fundamentals of Speech Recognition, Prentice Hall 1993.

[7] Pardo M., Sberveglieri G., "Learning from data: a tutorial with emphasis on modern pattern recognition methods", *IEEE Sensors Journal*, Vol.2 , No.3:203-217, 2002.

[8] Jain A. K., Duin R. P. W., Jianchang M., "Statistical pattern recognition: a review", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol.22 , No.1:4-37, 2000.

[9] Duda R. O., Hart P. E., Stork D. G., Pattern Classification, John Wiley & Sons, 2001.

[10] Vapnik V. N., The Nature of Statistical Learning Theory, Springer-Verlang, 1995.

[11] Biing-Hwang J., Furui S., "Automatic Recognition and Understanding of Spoken Language – A First Step Toward Natural Human-Machine Communication", *Proceedings of the IEEE*, Vol.88, No.8:1142:1165, 2000.

[12] Rabiner L. R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, Vol.77, No.2:257-286, 1989.

[13] Bilmes J.A., "What HMMs can do", U. Washington Tech Report, 2002.

[14] Bilmes J.A., "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models", Technical Report TR-97-021, ICSI, 1997.

[15] Bishiop C. M., Neural Networks for Pattern Recognition, Clarendon Press, Oxford, 1995.

[16] Morgan N., Bourlard H., "Continuous Speech Recognition: An introduction to the hybrid HMM/connectionists approach", *IEEE Signal Processing Magazine,* Vol.12, No.3:24-42, 1995.

[17] Bourlard H., Morgan N., Connectionist speech recognition: a hybrid approach, Kluwer Academic Publishers, 1994.

[18] Hayking S., Neural Networks: A Comprehensive Foundation, 2nd ed., Prentice Hall, 1999.

[19] Ganapathiraju A., Hamaker J.E., Picone J., "Applications of Support Vector Machines to Speech Recognition", *IEEE Transactions on Signal Processing*, Vol.52, No.8:2348-2355, 2004.

[20] Cristianini N., Shawe-Taylor J., Introduction to Support Vector Machines and other kernel-base learning methods, Cambridge University Press, 2000.

[21] Hermansky H., "Perceptual linear predictive (PLP) analysis of speech", *Journal of the Acoustical Society of America*, Vol.87, No.4:1738-1752, 1990.

[22] Hermansky H., Morgan N., "RASTA Processing of Speech", *IEEE Transactions on Speech and Audio Processing*, Vol.2, No.4:578-589, 1994.

[23] Gong Y., "Speech recognition in noisy environments: A survey", *Speech Communication*, Vol.16, No.3:261-291, 1995.

[24] Viikki O., Laurila K., "Cepstral domain segmental feature vector normalization for noise robust speech recognition", *Speech Communications*, Vol.25:133-147, 1998.

[25] Yoshizawa S., Hayasaka N., Wada N., Miyanaga Y., "Cepstral gain normalization for noise robust speech recognition", *Proceedings of ICASSP'04*, Vol.1:I209-212, 2004.

[26] Suk Y.H., Seung H.C., Hwang S.L, "Cepstrum third-order normalization method for noisy speech recognition", *IEEE Electronic Letters*, Vol.35, No.7:527-528, 1999.

[27] Hsu C.W, Lee L.S., "Higher order cepstral moment normalization (HOCMN) for robust speech recognition", *Proceedings of ICASSP'04*, Vol.1:I197-200, 2004.

[28] Strope B., Alwan A., "A Model of Dynamic Auditory Perception and Its Application to Robust Word Recognition", *IEEE Transactions on Speech and Audio Processing*, Vol.5, No.5:451-464, 1997.

[29] Furui S., "On the use of hierarchical spectral dynamics in speech recognition", *IEEE Proceedings of ICASSP'90*, Vol.2:3-6, 1990.

[30] Wan E. A., Nelson A. T., "Dual Extended Kalman Filter Methods", Kalman Filtering and Neural Networks, ed. Simon Haykin, Wiley & Sons, 2001.

[31] DeVaul R. W., "Creating a Gaussian Mixture Model using BNT – A Short Tutorial", http://www.media.mit.edu/wearables/mithril/BNT/mixtureBNT.txt.

[32] Murphy K., "A Brief Introduction to Graphical Models and Bayesian Networks", http://www.cs.ubc.ca/~murphyk/Bayes/bayes.html, 1998.

[33] Jordan M., Learning in Graphical Models, Kluwer Academic Publishers, 1998.

[34] Ma N., Bouchard M., Goubran R. A., "Perceptual Kalman Filtering for Speech Enhancement in Colored Noise", *Proceeding of ICASSP'04*, Vol.1:I717-720, 2004.

[35] Gabrea, M., "Robust adaptive Kalman filtering-based speech enhancement algorithm", *Proceedings of ICASSP'04*, Vol.1:I301-304, 2004.

[36] Paliwal K.K., Atal B.S., "Vector Quantization of LPC Parameters at 24 bits/frame", *IEEE Transactions on Speech and Audio Processing*, Vol.1, No.1:3-14, 1993.

[37] Tourneret J.Y., "Statistical properties of line spectrum pairs", *Signal Processing*, Vol. 65, No.2:239-255, 1998.

[38] El-Maleh K., Samouelian A., Kabal P.,"Frame level noise classification in mobile environments", *Proceeding of ICASSP'99*, Vol.1:237-240, 1999.

[39] Misra H., Ikbal S., Bourlard H., Hermansky H., "Spectral Entropy based feature for robust ASR", *Proceeding of ICASSP'04*, Vol.1:I193-196, 2004.

[40] Cooke M., Green P., Josifovski L., Vizinho A., "Robust automatic speech recognition with missing and unreliable acoustic data", *Speech Communication*, Vol.34, No.3:267-289, 2001.

[41] Raj B., Seltzer M. L., Stern R. M., "Reconstruction of missing features for robust speech recognition", *Speech Communications*, Vol.42, No.4:275-296, 2004.

[42] Haykin S., Adaptive Filter Theory, 4th ed., Pearson Education Inc., 2002.

[43] Proakis J. G., Manolakis D.G., Digital Signal Processing: Principles, Algorithms and Applications, Prentice Hall, 1996.

[44] Kajita S., Itakura F., "Subband-Autocorrelation analysis and its application for speech recognition", *Proceeding of ICASSP'94*, Vol.2:193-196, 1994.

[45] Kajita S., Takeda K., Itakura F., "Spectral wieghting of SBCOR for noise robust speech recognition", *Proceeding of ICASSP'98*, Vol. 2:621-624, 1998.

[46] Seneff S., "A joint synchrony/mean-rate model of auditory speech processing", *Journal of Phonetics*, Vol. 16:55-76, 1988.

[47] Amin M. G., "A Frequency-domain LMC comb filter", *IEEE Transactions on Circuits and Systems,* Vol. 38, No 12:1573-1576, 1991.

[48] VOICEBOX: Speech Processing Toolbox for MATLAB, http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html

[49] Bayes Net Toolbox for Matlab, http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html

[50] PLP and RASTA in Matlab, http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat

INDEX

R

S

U

V